Journal of Global Optimization (2006) 35: 423–441
DOI 10.1007/s10898-005-5608-4

Converging Marriage in Honey-Bees Optimization and Application to Stochastic Dynamic Programming

HYEONG SOO CHANG[1,2]
[1]*Department of Computer Science and Engineering, Sogang University, Sinsoo-Dong 1, Mapo-Gu, 121-742, Korea (e-mail: hschang@sogang.ac.kr)*
[2]*Program of Integrated Biotechnology, Sogang University, Seoul, Korea*

(Received 11 May 2004; accepted in revised form 29 November 2005)

Abstract. In this paper, we first refine a recently proposed metaheuristic called "Marriage in Honey-Bees Optimization" (MBO) for solving combinatorial optimization problems with some modifications to formally show that MBO converges to the global optimum value. We then adapt MBO into an algorithm called "Honey-Bees Policy Iteration" (HBPI) for solving infinite horizon-discounted cost stochastic dynamic programming problems and show that HBPI also converges to the optimal value.

Key words: Honey-bees optimization, Markov decision process, Policy iteration, Stochastic dynamic programming, Swarm intelligence

1. Introduction

Swarm intelligence-based optimization for solving difficult non-stochastic combinatorial optimization problems has been successful in various settings [6, 10, 23], for example, by properly casting the social search behaviors of biological ant colonies [13–15] and the sociological behavior associated with bird flocking and fish schooling [16] into artificial models of stochastic search. The success is mainly due to the fact that the optimization methods via swarm intelligence have distinguished computational advantages of distributed "simple" task processing, self-organization, and distributed information utilization in forming a "social" opinion during optimum-seeking process. Recently, Abbass [1, 2] proposed a metaheuristic called "Marriage in Honey-Bees Optimization" (MBO) for solving non-stochastic combinatorial optimization problems inspired from the phylogenetic sociality in honey-bees (the mating process of honey-bees), and applied the metaheuristic to a class of propositional satisfiability problems and empirically showed that the new method improves other well-known algorithms for the satisfiability problems. However, some parts of the algorithm description in [1, 2] are not clearly presented and more importantly, the problem to prove the convergence of the MBO algorithm has not been still resolved.

In this paper, we first refine MBO for solving non-stochastic combinatorial optimization problems to formally show that MBO converges to the global optimum value. We then adapt MBO into an algorithm called "Honey-Bees Policy Iteration" (HBPI) for solving *stochastic sequential decision-making* problems and show that HBPI also converges to the optimal value.

The artificial model for the social behaviors of honey-bees in the present paper is the *monogynous* colony that has a *single* queen, for simplicity. Extensions to *polygynous* colony having more than one queen is straightforward. The model in the present paper is in the context of stochastic search for optimization, not exactly imitating the behaviors of the real honey-bees colonies [12], based on the models considered in the MBO algorithms [1, 2].

The honey-bees colony in our MBO setting consists of a queen, drones, workers, and broods. The queen takes the charge of reproduction of new individuals (potential solutions to a given problem). Drones are haploid (having one copy of the allele to determine the sex or having half the genetic material of a worker) individuals and represent the fathers of the colony. Workers are responsible for broods care. The honey-bees colony basically operates as follows. Mating begins with a waggle dance performed by the queen. The queen takes off on her mating flights followed by the drones. Mating takes place in the air where the queen mates a number of (probabilistically) selected drones in each flight. Sperms from the selected drones are added and accumulated in the spermatheca of the queen to form the genetic pool of potential broods ("premature" potential solutions of the problem). For every fertilized egg laid by the queen, sperm is retrieved randomly from her spermatheca and the sperm is "mixed" with the queen's genotype in order for the queen to breed. The newly born broods are then fostered by the workers (improving the premature solutions by a heuristic local search), and finally among the fostered broods and the current queen, a new queen is elected. The sequence of the newly selected queen in MBO preserves the monotonicity property and MBO converges to the global optimum value (the global optimum if it is unique) to the given problem with probability one.

MBO is then adapted to HBPI algorithm for approximately solving infinite horizon discounted cost stochastic dynamic programming (SDP) problems, also known as Markov Decision Processes (MDPs) [4, 24], extending the applicability of the MBO metaheuristic into stochastic sequential decision making problems. Unfortunately, a stochastic control problem formulated by SDP often experiences *the curse of dimensionality*, which makes the application of the well-known exact algorithms, e.g., value iteration (VI) or policy iteration (PI) [24], impractical (see Section 3).

HBPI inherits the spirit of PI that solves MDPs exactly. Contrary to PI, by directly manipulating the policies, HBPI eliminates the operation of minimization over the entire action space in the "policy improvement" step of PI. HBPI preserves a certain monotonicity property via "elitist" policies and converges to the optimal value with probability one.

This paper is organized as follows. We start with presenting the MBO algorithm and analyze its convergence in Section 2. In Section 3, we describe and analyze HBPI. We conclude the paper with some remarks in Section 4.

2. Honey-Bees Optimization

2.1. ALGORITHM DESCRIPTION

Consider an optimization problem of

$$\min_{x\in\Omega} f(x),$$

where $\Omega=\{0,1\}^n$ is a finite solution space of binary strings (or genotypes) of the length $n<\infty$ and $f\colon \Omega\to\mathcal{R}^+$ is a nonnegative cost function. Our goal is to find $x^*\in\Omega$ that achieves the minimum. A high-level description of MBO is shown in Figure 1, where some steps are described at a conceptual level, with details provided below.

The algorithm starts with initializing the queen's genotype $q_0\in\Omega$ arbitrarily and the size of the queen's spermatheca $M\geqslant 1$ that denotes the maximum possible number of matings (sperms of drones; we will use the terms of "sperm of drone" and "drone" interchangeably) that can be deposited per the queen's mating flight (the first two steps of the *Queen's Mating Flight Initialization* step and the inner *Repeat* step in the outer *Repeat* step in Figure 1). The total number of the broods $j\geqslant 1$ to be created in the *Broods Generation and Improvements* step needs to be selected and the mutation probability $\mu\in(0,1)$ for the mutation of the broods needs to be selected, too. The value of the initial speed $s\in(0,1]$, the minimum energy $E_{\min}\in(0,1]$, the initial energy $e\in(E_{\min},1]$, the discount factor $\alpha\in(0,1]$ for the speed and the reduction factor β for the energy such that $\lceil\frac{e-E_{\min}}{\beta}\rceil\geqslant 1$ need to be also initialized for each queen's mating flight, respectively.

The algorithm then iteratively improves the queen's genotype. From the *New Queen Generation* step, it can be easily seen that the following monotonicity is forced by the algorithm:

$$f(q_{m+1})\leqslant f(q_m),\quad m=0,1,\ldots$$

Therefore, this monotonicity implies that once q_m is equal to an optimal solution $x^*\in\Omega$, the algorithm converges to the global optimum value in the sense that $f(q_m)=f(q_i)=\min_{x\in\Omega} f(x)$ for all $i>m$.

```
Marriage in Honey-Bees Optimization (MBO)

• Initialization:
  Select the size of the spermatheca $M \geq 1$. Initialize $q_0 \in \Omega$ arbitrarily.
  Set $m = 0$ and select $\mu \in (0,1)$, $e \in (E_{\min}, 1]$, $s \in (0,1]$, and $j \geq 1$.
  Set $\beta$ such that $\lceil \frac{e - E_{\min}}{\beta} \rceil \geq 1$ and $\alpha \in (0,1]$.
• Repeat:
  – Queen's Mating Flight Initialization:
    Generate $t_{-1}$ arbitrarily. Set $k = 0$ and $P = \emptyset$. $E(0) = e$ and $S(0) = s$.
  – Repeat while $E(k) \geq E_{\min}$:
    * Potential Drone Selection:
      Generate a random drone $t_k$ with $S(k)$.
    * Potential Drone Acceptance (Mating):
      With probability $\min\{1, \exp((f(t_{k-1}) - f(t_k))/S(k))\}$, add $t_k$'s sperm
      to queen's spermatheca: $P \leftarrow P \cup \{t_k\}$.
      If $|P| > M$, $P \leftarrow P - \{t\}$ where $t \in \arg\max_{x \in P} f(x)$.
    * $E(k+1) = E(k) - \beta$, $S(k+1) = \alpha \cdot S(k)$, and $k \leftarrow k + 1$.
  – Broods Generation and Improvements:
    * Obtain the elitist drone from $P$: $b_e \in \arg\min_{x \in P} f(x)$.
    * Select $b_1, ..., b_j$ in $P$ and crossover $b_i$ with $q_m$ generating $B_c = \{b_1^c, ..., b_j^c\}$.
    * Mutate each brood in $B_c$ with $\mu$ generating $B_m = \{b_1^m, ..., b_j^m\}$.
    * Improve each brood in $B_m$ via workers generating $B_w = \{b_1^w, ..., b_j^w\}$.
    * Update the elitist drone $b_e^u \leftarrow \arg\min_{x \in \{b_e\} \cup B_w} f(x)$.
  – New Queen Generation: $q_{m+1} \in \arg\min_{x \in \{q_m, b_e^u\}} f(x)$.
  – $m \leftarrow m + 1$.
```

Figure 1. Honey-bees optimization.

At each iteration of $m \geqslant 0$, the queen starts a mating flight. To ensure that the queen flies for a certain amount of time, the initial values of the energy $E(0)$ and the speed $S(0)$ of the queen are initialized with e and s, respectively and reduced by decrement of β and factor of α, respectively:

$$E(k+1) = E(k) - \beta, \quad S(k+1) = \alpha \cdot S(k), \ \ \alpha \in (0, 1],$$

until the value of the energy is no larger than the predetermined value of $E_{\min}$. In other words, the total number of flies in terms of iteration k by the queen is $\lceil \frac{e - E_{\min}}{\beta} \rceil$. The energy serves as a "counter" controlled by the parameter β. As mentioned before, this *emulates* the queen's mating flight time, i.e., controls the degree of a local search in the solution space. We follow here the original structure/algorithm description of MBO even though there would be much simpler way of describing it.

At each fly of the queen, a new potential drone (new genotype) $t_k \in \Omega$ for mating with the queen is generated by probabilistically flipping each bit independently with probability $S(k)$ in the current drone's genotype t_{k-1} with the probability of

$$S(k)^{H(t_{k-1}, t_k)}(1 - S(k))^{n - H(t_{k-1}, t_k)}, \tag{1}$$

where $H(\cdot)$ is the Hamming distance between t_{k-1} and t_k and the speed $S(k)$ of the queen acts as the probability of changing the current bit value at each bit position (the *Potential Drone Selection* step in Figure 1). If the resultant new genotype has a lower cost value $(f(t_{k-1}) > f(t_k))$, then it is accepted with the probability 1. If the new genotype has a higher cost value $(f(t_{k-1}) \leqslant f(t_k))$, then it will be accepted with certain probability given by the Boltzmann factor. That is, with probability

$$\min\left\{1, \exp\left(\frac{f(t_{k-1}) - f(t_k)}{S(k)}\right)\right\},$$

t_k's sperm is added to the queen's spermatheca: $P \leftarrow P \cup \{t_k\}$ (the *Potential Drone Acceptance* step in Figure 1). Note that the set P is a *multiset*. As in Simulated Annealing (SA) [21], this enables the algorithm to hill-down from a locally optimal solution.

The queen can hold at most M potential drones. So, if $|P| > M$,

$$P \leftarrow P - \{t\} \quad \text{where } t \in \arg\max_{x \in P} f(x).$$

In other words, if the size of $P > M$, then we select one sperm of $\arg\max_{x \in P} f(x)$ and remove it from P. This step is different from the original MBO algorithms presented i [1, 2]. To the author's understanding, in the original MBO in [1, 2], the comparison between the most recently added drone and t_k is made and then t_k is added as a potential drone "probabilistically" if t_k is worse than the most recently added drone. t_k replaces the most recently added drone if t_k is better between the two. On the other hand, here we eliminate the worst genotype from the spermatheca while maintaining the total number of drones in the spermatheca by M. Observe that the speed of the queen acts similarly to the temperature parameter in SA, making the probability of having a large change of the current genotype in the search space decrease gradually.

Once the queen's mating flights are finished, the queen returns to the nest and the brood creation process begins (the *Broods Generation and Improvements* step in Figure 1). This process is very similar to Genetic Algorithm (GA) [25, 28]. We first keep the elitist sperm from the queen's spermatheca P:

$$b_e \in \arg\min_{x\in P} f(x).$$

We then select j sperms, $b_1, \dots, b_j$ from P with the equal probability of $1/|P|$ or by the cost proportional probability; b_i is selected with the probability of $f(b_i)^{-1}/\sum_{b\in P} f(b)^{-1}$ (if f is a positive function) and then apply the crossover operation to each b_i with the current genotype of the queen q_m, generating the brood set $B_c = \{b_1^c, \dots, b_j^c\}$. The crossover operation method can be the single-point, the two-point, or the uniform types [27], even an adaptive crossover with tuning crossover probability (see, e.g., [28]), etc. Each $b_i^c \in B_c$ is subsequently mutated to b_i^m with the mutation probability of $\mu \in (0, 1)$:

$$\mu^{H(b_i^c, b_i^m)}(1-\mu)^{n-H(b_i^c, b_i^m)},$$

where μ is the probability of changing the bit value independently at each bit position. The mutation process creates the mutated brood set $B_m = \{b_1^m, \dots, b_j^m\}$. Each brood in B_m is then improved by the workers, generating the improved set $B_w = \{b_1^w, \dots, b_j^w\}$. The workers corresponds to a local search algorithm, which is similar to the idea of memetic algorithm [10] that incorporates a local search in GA. We update the elitist brood by

$$b_e^u \leftarrow \arg\min_{x\in\{b_e\}\cup B_w} f(x).$$

Finally, the queen's genotype is updated by

$$q_{m+1} \in \arg\min_{x\in\{q_m, b_e^u\}} f(x).$$

REMARK. For the potential drone acceptance probability given by $\min\{1, \exp((f(t_{k-1}) - f(t_k))/S(k))\}$, we can replace t_{k-1} with q_m as in the original MBO [1]. Then, the queen accepts t_k as a potential drone in her spermatheca even if t_k has the higher cost value and the probability of acceptance will be high when the cost of the drone t_k is as good as the queen's cost or when the queen's speed is still high. The high speed gives more emphasis on exploration in the search space than exploitation and the low speed gives more emphasis on exploitation. Even if q_m affects the generation of q_{m+1}, the mutation process of the *Broods Generation and Improvements* step is independently processed with the *Repeat* step. It can be shown that MBO still converges to the global optimum value with a slight change to the proof of Theorem 2.1. In this case, we can alternatively use the proof techniques from the convergence results in the Ant Colony Optimization literature, e.g., Gutjahr's convergence analysis [17, 18] to handle the convergence.

MBO can be (roughly) viewed as a hybrid scheme that combines SA and GA with some differences, where SA part corresponds to the queen's mating flight to obtain the potential drone policies' sperms in her spermatheca and GA part corresponds to *Broods Generation and Improvements* step with some differences.

SA is based on the Metropolis scheme. An initial state of a thermodynamic system is chosen at some energy (the cost value of a solution for a given problem) and temperature. Holding the temperature fixed, the initial configuration is perturbed and the change in energy is computed. If the new configuration is better, it is accepted. If not, it is accepted with a probability given by the Boltzmann factor. This local search behavior for a fixed temperature is then repeated for a sufficient time to achieve a certain equilibrium, and then the temperature is decremented and the entire process repeated until a frozen state is achieved (see, e.g., [21]). The potential drone policy acceptance rule of MBO is exactly the same as SA's. However, MBO does not repeat the local search for a "long" time for each fixed queen's speed (temperature). The local search is done exactly once for each speed with the control of the sperm contents of her spermatheca by keeping only the best M drones generated so far until the queen's energy gets worn out. Furthermore, the perturbation of the solution is done with the queen's speed, which emulates the queen's position changes, correspondingly a drone's fly that follows the queen's mating flight. (Of course, we can actually incorporate the exact SA procedures in MBO instead of the procedure given in Figure 1, but then losing the spirit of the social behavior of the honey-bees colony. In fact, if we set $\alpha = 1$, then the queen's mating flight part corresponds to the repeated local search in SA with a fixed temperature, where the local search duration is controlled with $E(k)$, the energy of the queen.)

GA is based on the three basic operations of selection, crossover, and mutation over the populations. The *Broods Generation and Improvements* step operates the three basic procedures in GA with incorporation of the elitist strategy by De Jong [11]. In contrast to GA, the honey-bees colony has the workers that foster the premature broods. The local search used in MBO depends on the problem being solved, for example, 2-opt and 3-opt algorithms for traveling salesman problems [3, 19].

2.2. CONVERGENCE ANALYSIS

THEOREM 2.1. *MBO converges to the global optimum value in the sense that*

$$\lim_{m\to\infty} \Pr\left\{q_m \in \arg\min_{x\in\Omega} f(x)\right\} = 1.$$

Proof. Let us first fix the iteration $m \geqslant 0$ arbitrarily. From the choice of $e = E(0)$, for $k > \frac{e-E_{\min}}{\beta}$, $E(k) < E_{\min}$ so that the inner *Repeat* step finishes with $k = \hat{k} = \lceil \frac{E(0)-E_{\min}}{\beta} \rceil$.

From the inner *Repeat* step in Figure 1, once an optimal solution $x^* \in \arg\min_{x\in\Omega} f(x)$ is generated, it is deposited into the queen's spermatheca with probability 1 and it is kept there throughout the *Repeat* step. Let the minimal probability of generating x^* in the *Repeat* step p given as

$$p = \min_{k=1,\dots,\hat{k}-1} \min_{x\in\Omega} \left\{ S(k)^{H(x,x^*)} (1 - S(k))^{n-H(x,x^*)} \right\},$$

where $H(x, x^*)$ denotes the Hamming distance between genotypes x and x^*. Observe that p is in $(0, 1)$ because $S_{\min} = S(0)\alpha^{\hat{k}}$ with $\alpha \in (0, 1]$ is in $(0, 1)$.

We then have that

$$\Pr\{t_{\hat{k}} = x^*\} \geqslant 1 - (1-p)^{\hat{k}} > 0,$$

which immediately implies that

$$\Pr\{b_e = x^*\} \geqslant 1 - (1-p)^{\hat{k}}.$$

Now let the minimal probability of mutating into x^* or generating x^*

$$p_b = \min_{x\in\Omega} \left\{ \mu^{H(x,x^*)} (1-\mu)^{n-H(x,x^*)} \right\} \in (0, 1)$$

($p_b = \mu^n$ for $\mu < 1/2$). Then the probability that q_{m+1} is x^* is lower bounded by

$$\begin{aligned}\Pr\{q_{m+1}=x^*\} \geqslant \Pr\{b_e^u=x^*\} &\geqslant \Pr\{b_e=x^*\}\\ &\quad+\Pr\{x^*\in B_w\}-\Pr\{b_e=x^* \text{ and } x^*\in B_w\}\\ &\geqslant (1-(1-p)^{\hat{k}})+(1-(1-p_b)^j)\\ &\quad-(1-(1-p)^{\hat{k}})(1-(1-p_b)^j)\\ &>0.\end{aligned}$$

The above result holds for any fixed $m=0,1,2,\ldots$. Therefore, with some $\epsilon\in(0,1]$ and the selection rule of

$$q_{m+1}\in\arg\min_{x\in\{q_m,b_e^u\}} f(x),$$

we have that for any fixed $m\geqslant 1$,

$$\Pr\{q_m=x^*\}\geqslant 1-(1-\epsilon)^m>0,$$

which proves the statement of Theorem. $\square$

REMARK 1. From the result of Theorem 2.1, to obtain a “good” probability bound on the performance of MBO, the size of the broods populations, i.e., the value of j and the size of $\hat{k}$ might need to be large for applying MBO in practice. However, the analysis here is based on a very coarse estimation of the probabilities that the potential drone selection and the mutation processes generate an optimal solution so that convergence to the optimum value with high probability may be achieved with a moderate size of the parameters. Furthermore, from the monotonicity property of the sequence $\{q_m\}$, if the global optimum $x^*\in\Omega$ is unique, MBO converges to the global optimum with probability 1.

REMARK 2. As in convergence analysis of evolutionary algorithms in other works (see, e.g., [26]), the proof of Theorem 2.1 can be done with a Markov chain analysis but our proof is simpler.

3. Application to Stochastic Dynamic Programming

In this section, we adapt MBO into an algorithm for solving infinite horizon discounted cost stochastic dynamic programming problems (see, e.g, [4, 24] for a substantial discussion) with suitable extensions and modifications of MBO and show that the algorithm converges to an optimal solution of a given SDP problem. We begin with some backgrounds.

3.1. PRELIMINARIES

Consider an infinite horizon discounted cost SDP problem (X, A, P, C) with finite state space X, finite action space A, cost function C such that $C\colon X \times A \rightarrow \mathcal{R}$, and transition function P that maps $\{(x, a) | x \in X, a \in A\}$ to the set of all possible probability distributions over X. We denote the probability of transitioning to state $y \in X$ when taking action a in state $x \in X$ by $P(x, a)(y)$. For simplicity, we assume that every action is admissible in every state.

Let Π be the set of all stationary policies $\pi : X \rightarrow A$. Define the *optimal value* associated with an initial state $x \in X$:

$$V^*(x) = \min_{\pi \in \Pi} V^{\pi}(x), x \in X, \text{ where}$$
$$V^{\pi}(x) = E\left[\sum_{t=0}^{\infty} \gamma^t C(x_t, \pi(x_t)) \middle| x_0 = x\right], \quad x \in X, \ 0 < \gamma < 1, \ \pi \in \Pi,$$

where x_t is a random variable denoting state at time t and γ is the discount factor. The sequence $\{x_t, t \geqslant 0\}$ is a stochastic process defined by following π. Fixing a policy $\pi \in \Pi$ induces a Markov chain. Throughout the remaining of paper, we assume that γ is fixed.

Given a fixed initial state probability distribution δ defined over X, we define the *average value of* π *for* δ or *cost value of* π:

$$J_{\delta}^{\pi} = \sum_{x \in X} V^{\pi}(x)\delta(x).$$

The problem is to find an optimal policy $\pi^* \in \Pi$ that achieves

$$J_{\delta}^{*} = \min_{\pi \in \Pi} J_{\delta}^{\pi}.$$

Unfortunately, many stochastic control problems formulated by SDP experience the curse of dimensionality, which makes the application of VI or PI [24], impractical. The time-complexity depends on the size of the state space and on the size of the action space, so that the runtime can easily get prohibitive if these sizes are large. The running time complexity of VI is polynomial in $|X|$, $|A|$, and $1/(1-\gamma)$ and in particular one iteration takes $O(|X|^2|A|)$ time. For PI, doing the policy improvement step takes $O(|X|^2|A|)$ time. See, e.g., [22] for a detailed discussion including the state and action space-dependent time complexity of the linear programming approach for solving MDPs. Therefore, applying the exact methods for solving MDPs is very difficult if the state and/or the action space are large, which is true for many interesting problems.

Numerous approximation-based schemes exist to get away with the dimensionality problem via various techniques, e.g., structural analysis, aggregation, sampling, feature extraction, etc. (see, e.g., [5, 24, 29]). However, there are few algorithms that incorporate the successful techniques of *evolutionary computation* into the SDP context. Exceptions are recent works by Chang et al. [8, 9] based on GA and an ant system. A comparison study of these evolutionary computation-based approaches for solving SDP problems is a good research topic but beyond the present paper's scope.

Even though a single iteration time complexity $O(|X|^2|A|)$ of VI is smaller than $O(|X|^2|A|+|X|^3)$-complexity of PI, the number of iterations required for VI can grow *exponentially* in the discount factor. Indeed, it can be shown that PI converges faster to the optimal value than VI in terms of the number of iterations if both algorithms begin with the same value [24] and it is known that VI is usually outperformed by PI in practical applications [20]. Furthermore, the structure of MBO naturally induces a PI-like algorithm with an adaptation of MBO. Therefore, we review the PI algorithm briefly here. See, e.g., [24] for a detailed discussion.

PI computes π^* in a finite number of steps because there are a finite number of policies in Π and PI preserves the monotonicity in terms of the policy performance. The PI algorithm consists of two parts: policy evaluation and policy improvement. Let $B(X)$ be the space of real-valued functions on X. We define an operator T: $B(X) \to B(X)$ as

$$T(\Phi)(x)=\min_{a\in A}\left\{C(x,a)+\gamma\sum_{y\in X}P(x,a)(y)\Phi(y)\right\}, \quad \Phi\in B(X), \quad x\in X \quad (2)$$

and similarly, an operator $T_\pi : B(X)\to B(X)$ for $\pi\in\Pi$ as

$$T_\pi(\Phi)(x)=C(x,\pi(x))+\gamma\sum_{y\in X}P(x,\pi(x))(y)\Phi(y), \quad \Phi\in B(X), \quad x\in X. \quad (3)$$

For each policy $\pi\in\Pi$, there exists a corresponding unique $\Phi\in B(X)$ such that for $x\in X$,

$$T_\pi(\Phi)(x)=\Phi(x) \quad \text{and} \quad \Phi(x)=V^\pi(x).$$

The policy evaluation step obtains V^π for a given π, and the policy improvement step obtains $\hat{\pi}\in\Pi$ such that

$$T(V^\pi)(x)=T_{\hat{\pi}}(V^\pi)(x), \quad x\in X.$$

The policy $\hat{\pi}$ improves π in that $V^{\hat{\pi}}(x)\leqslant V^\pi(x)$ for all $x\in X$, implying that for any δ,

$$J_\delta^{\hat{\pi}}\leqslant J_\delta^\pi.$$

3.2. HONEY-BEES POLICY ITERATION

HBPI is inspired from PI and is adapted from MBO within the context of SDP with a key operation called "policy switching" [7], which is much like crossover operation in GA, for multiple policies manipulation. HBPI starts with an arbitrarily selected initial queen's policy $\pi_0 \in \Pi$ and at each iteration $m \geqslant 0$, HBPI generates π_{m+1}, which is no worse than the current policy π_m, i.e., $J_\delta^{\pi_{m+1}} \leqslant J_\delta^{\pi_m}$. It is shown below that the sequence of the policies $\pi_m, m = 0, 1, 2, \ldots$ converges to π^* with probability one. As in MBO, we need to initialize and select some parameters (refer the *Initialization* step in Figure 2). In addition, we introduce more parameters: we denote P_m as the mutation selection probability, P_g the global mutation probability, and P_l the local mutation probability. We also define *action selection distribution* ξ as a probability distribution over A such that $\sum_{a \in A} \xi(a) = 1$ and $\xi(a) > 0$ for all $a \in A$.

A high-level description of HBPI is shown in Figure 2, where some steps are described at a conceptual level, with details provided in the following subsections. Because overall procedure of HBPI is similar to MBO, we provide details only for necessary parts.

3.2.1. *Queen's Mating Flight*

The queen's mating flight portion is similar to MBO's in Figure 1. We need to select/generate a potential drone policy for mating with the queen. In MBO, the speed of the queen, $S(k)$ was directly used as the probability of generating a potential drone's genotype (refer Equation (1)). Here we generate a random drone policy ϕ_k with $S(k)$, where $S(k)$ is used for the probability of *determining whether the old potential drone's policy is globally or locally transformed* in the *Potential Drone Policy Selection* step in Figure 2.

In the *Potential Drone Policy Selection* step in Figure 2, for each state of the (previously considered) drone's policy ϕ_{k-1}, the specified action is altered probabilistically. We distinguish between two types of transformation – "local" and "global" – which are differentiated by how much of the policy ϕ_{k-1} is likely be changed. Local transformation is intended to guide the algorithm in obtaining a true optimal policy through local search of "nearby" policies, whereas the global mutation allows HBPI to escape from local optima. A high transformation probability indicates that many components of the policy vector are likely to be modified, representing a more global change, whereas a low mutation probability implies that very little transformation is likely to occur, meaning a more localized perturbation. For this reason, we assume that $P_l \ll P_g$, with P_l being very close to zero and P_g being very close to one.

```
Honey-Bees Optimization for Stochastic Dynamic Programming

• Initialization:
    Select the size of the spermatheca $M \geq 1$. Initialize $\pi_0 \in \Pi$ arbitrarily.
    Set $m = 0$ and select $e \in (E_{\min}, 1]$, $s \in (0, 1)$, and $j \geq 1$. Select $P_m, P_g, P_l \in (0, 1)$.
    Set $\beta$ such that $\lceil \frac{e - E_{\min}}{\beta} \rceil \geq 1$ and $\alpha \in (0, 1]$.
• Repeat:
    – Queen's Mating Flight Initialization:
        Generate $\phi_{-1}$ arbitrarily. Set $k = 0$ and $P = \emptyset$. $E(0) = e$ and $S(0) = s$.
    – Repeat while $E(k) \geq E_{\min}$:
        * Potential Drone Policy Selection:
            Generate a random drone policy $\phi_k$ by transforming $\phi_{k-1}$ globally
            with $S(k)$ using $P_g$ and $\xi$ or locally with $1 - S(k)$ using $P_l$ and $\xi$.
        * Potential Drone Policy Acceptance (Mating):
            With probability $\min\{1, \exp((J_\delta^{\phi_{k\;1}} - J_\delta^{\phi_k})/S(k))\}$, add $\phi_k$'s sperm
            to queen's spermatheca: $P \leftarrow P \cup \{\phi_k\}$.
            If $|P| > M$, $P \leftarrow P - \{t\}$ where $t \in \arg\max_{\pi \in P} J_\delta^\pi$.
        * $E(k+1) = E(k) - \beta$, $S(k+1) = \alpha \cdot S(k)$, and $k \leftarrow k + 1$.
    – Broods Generation and Improvements:
        * Policy Switching:
            · Obtain the elitist drone policy from $P$: $\phi_e \in \arg\min_{\pi \subset P} J_\delta^\pi$.
            · Select $\phi_1, ..., \phi_j$ in $P$ and generate $\{\phi_1^c, ..., \phi_j^c\}$, where
                  $\phi_i^c(x) \in \{\arg\min_{\pi \in \{\phi_i, \pi_m\}} (V^\pi(x))(x)\}, x \in X.$
        * Policy Mutation: For each policy $\phi_i^c, i = 1, ..., j$,
            · Generate a globally mutated policy $\phi_i^m$ with $P_m$ using $P_g$ and $\xi$
              or a locally mutated policy $\phi_i^m$ with $1 - P_m$ using $P_l$ and $\xi$.
        * Update the elitist $\phi_e^u(x) \in \{\arg\min_{\pi \in \{\phi_e, \phi_1^m, ..., \phi_j^m\}} (V^\pi(x))(x)\}, x \in X$.
    – New Queen Generation: $\pi_{m+1}(x) \in \{\arg\min_{\pi \in \{\pi_m, \phi_e^u\}} (V^\pi(x))(x)\}, x \in X$.
    – $m \leftarrow m + 1$
```

Figure 2. Honey-bees optimization for stochastic dynamic programming.

We first determine whether the transformation will be global or local. The transformation is global with probability $S(k)$ and local otherwise. If the policy ϕ_{k-1} is globally (locally) mutated, for each state x, $\phi_{k-1}(x)$ is changed with probability $P_g(P_l)$. If a transformation does occur, it is carried out according to the action selection distribution ξ, i.e., the new potential drone's policy ϕ_k is generated according to $\Pr\{\phi_k(x)=a\}=\xi(a)$, for all modified states x (the actions for all other states remain unchanged). For example, one simple ξ is the uniform action selection distribution, in which case the new (modified) policy would randomly select a new action for each modified state (independently) with equal probability over the entire action space.

Note that because the value of the queen's speed gets smaller in the inner *Repeat* step, the probability that a local transformation will occur gets higher. This is also similarly reflected in the *Potential Drone Policy Acceptance* step as in MBO.

3.2.2. *Policy Switching*

A key operation in HBPI is policy switching that combines multiple policies automatically to generate an *improved policy* no worse than all of the policies that policy switching combines. The policy switching operation is in spirit of crossover operation in GA and at the same time the workers' broods improvements in MBO because the combined policy is guaranteed to be an improved policy. Note that this property does not hold in general for combinatorial optimization methods so that a *heuristic* local search is usually used for (broods) improvements compared with a systematic improvement method of policy switching.

Formally, given a nonempty subset Δ of Π, we define a policy $\bar{\pi}$ generated by *policy switching* with respect to Δ as

$$\bar{\pi}(x) \in \left\{ \arg\min_{\pi\in\Delta}(V^{\pi}(x))(x) \right\}, \quad x \in X. \tag{4}$$

The following result states that $\bar{\pi}$ improves all policies in Δ. See [7] for a proof.

THEOREM 3.1. Consider a nonempty subset Δ of Π and the policy $\bar{\pi}$ generated by policy switching with respect to Δ given in Equation (4). Then, for all $x \in X$,

$$V^{\bar{\pi}}(x) \leqslant \min_{\pi\in\Delta} V^{\pi}(x).$$

The above theorem immediately implies the following result (see Corollary 3.1 in [9]):

COROLLARY 3.1. Consider a nonempty subset Δ of Π and the policy $\bar{\pi}$ generated by policy switching with respect to Δ given in Equation (4). Then, for any initial state distribution δ,

$$J_\delta^{\bar{\pi}} \leqslant \min_{\pi \in \Delta} J_\delta^\pi .$$

From the above results and from the construction of π_m given as

$$\pi_{m+1}(x) \in \left\{ \arg\min_{\pi \in \{\pi_m, \phi_e^u\}} (V^\pi(x))(x) \right\}, \quad x \in X,$$

we have the following:

COROLLARY 3.2. For any δ and for all $m \geqslant 0$,

$$J_\delta^{\pi_{m+1}} \leqslant \min \left\{ J_\delta^{\pi_m}, J_\delta^{\phi_e^u} \right\}.$$

In other words, the new queen's policy π_{m+1} for the iteration $m+1$ improves any policy generated over the iteration m (i.e., any policy in the queen's spermatheca P before the *Broods Generation and Improvements* step and $\phi_i^c, \phi_i^m, i = 1, \ldots, j$) and the queen's policy π_m at the iteration m. In this sense, we can view HBPI as a variant of the PI algorithm with "multi-policy" improvement.

Note that policy switching directly manipulates policies to generate an improved policy relative to all policies it was applied to, eliminating the operation of minimization over the entire action space. The computational time complexity of policy switching is $O(r|X|)$, where r is the number of policies policy switching operates on and the complexity is independent of the action space size. This is the main computational advantage that replaces the policy improvement step in the original PI for the SDP problems with large action spaces.

REMARK 1. Suppose that the action space A is relatively small, making the direct manipulation of policies by policy switching not helpful. Then, we can replace the *Policy Switching* step with a method called "parallel rollout" [7], which is a generalization of the policy improvement step of PI with *multi-policy* improvement. We simply replace policy switching operation with

$$\bar{\pi}(x) \in \arg\min_{a \in A} \left\{ C(x,a) + \gamma \sum_{y \in X} P(x,a)(y) \min_{\pi \in \Delta} V^\pi(y) \right\}, \quad x \in X$$

to generate a new policy $\bar{\pi}$ with respect to a set of policies Δ. Then every property we discussed before still holds and it is guaranteed that the resulting HBPI is faster than PI in terms of the number of iterations (with the same initialization for both algorithms).

REMARK 2. The elitist concept used in HBPI is different from the one used in MBO from De Jong's [11]. The elitist for a set $B \subseteq \Omega$ in MBO is $\text{argmin}_{x \in B} f(x)$. On the other hand, the elitist for a set $\Delta \subseteq \Pi$ in HBPI is a policy that is no worse than any $\pi \in \Delta$.

3.2.3. *Policy Mutation*

Policy mutation takes a given policy, and for each state, alters the specified action probabilistically as in the *Potential Drone Policy Selection* step. The main reason to generate mutated policies is to avoid being caught in a local minimum, making a probabilistic convergence guarantee possible.

As before, we distinguish between two types of mutation – local and global – The *Policy Mutation* step in Figure 2 first determines whether the mutation will be global or local, with probability P_m. If the policy π is globally (locally) mutated, for each state x, $\pi(x)$ is changed with probability $P_g(P_l)$. If a mutation does occur, it is carried out according to the action selection distribution ξ.

3.2.4. *Convergence Analysis*

THEOREM 3.2. Given P_m, P_g, and $P_l \in (0, 1)$ and an action selection distribution ξ such that $\sum_{a \in A} \xi(a) = 1$ and $\xi(a) > 0$ for all $a \in A$, as $m \to \infty$, $V^{\pi_m}(x) \to V^*(x), x \in X$ with probability one uniformly over X, regardless of π_0.

Proof. The proof is very similar to the proof of Theorem 2.1 with small extensions. As before, let us first fix the iteration $m \geqslant 0$ arbitrarily. From the choice of $e = E(0)$, for $k > \frac{e - E_{\min}}{\beta}$, $E(k) < E_{\min}$ so that the inner *Repeat* step in Figure 2 finishes with $k = \hat{k} = \lceil \frac{E(0) - E_{\min}}{\beta} \rceil$.

From the inner *Repeat* step, once an optimal policy π^* is generated, it is deposited into the queen's spermatheca with probability 1 and it is kept there throughout the inner *Repeat* step.

Let θ_l be the probability of generating one of the optimal policies by local transformation and let θ_g the probability of generating one of the optimal policies by global transformation. Then,

$$\theta_l \geqslant \prod_{x \in X} P_l \xi(\pi^*(x)) = (P_l)^{|X|} \cdot \prod_{x \in X} \xi(\pi^*(x)) > 0$$

$$\theta_g \geqslant \prod_{x \in X} P_g \xi(\pi^*(x)) = (P_g)^{|X|} \cdot \prod_{x \in X} \xi(\pi^*(x)) > 0, \tag{5}$$

where π^* is a particular optimal policy in Π that achieves $V^*(x), x \in X$. Then the minimal probability p of generating π^* in the inner *Repeat* step is given by

$$p = \min_{k=1,\dots,\hat{k}-1} \left\{ S(k)\theta_g + (1 - S(k))\theta_l \right\} > 0.$$

We then have that

$$\Pr\{\phi_{\hat{k}} = \pi^*\} \geqslant 1 - (1-p)^{\hat{k}} > 0,$$

which immediately implies that

$$\Pr\{\phi_e = \pi^*\} \geqslant 1 - (1-p)^{\hat{k}},$$

which further implies that because

$$\pi_{m+1}(x) \in \left\{ \arg\min_{\pi \in \{\pi_m, \phi_e^u\}} (V^\pi(x))(x) \right\}, \quad x \in X,$$

$$\Pr\{V^{\pi_{m+1}}(x) \leqslant V^*(x), \forall x \in X\} \geqslant 1 - (1-p)^{\hat{k}}.$$

Let the probability of generating π^* by the mutation process p_b in the *Policy Mutation* step:

$$p_b \geqslant P_m (P_g)^{|X|} \cdot \prod_{x \in X} \xi(\pi^*(x)) + (1 - P_m)(P_l)^{|X|} \cdot \prod_{x \in X} \xi(\pi^*(x)).$$

Then we have that

$$\begin{aligned}\Pr\{V^{\pi_{m+1}}(x) \leqslant V^*(x), \forall x \in X\} \geqslant & (1 - (1-p)^{\hat{k}}) + (1 - (1-p_b)^j) \\ & - (1 - (1-p)^{\hat{k}})(1 - (1-p_b)^j) > 0.\end{aligned}$$

This result holds for any fixed $m = 0, 1, 2, \dots$. Therefore, with some $\epsilon \in (0, 1]$,

$$\Pr\left\{V^{\pi^m}(x) = V^*(x), \forall x \in X\right\} \geqslant 1 - (1-\epsilon)^m > 0,$$

which proves the claim. □

4. Conclusion

In this paper, we refined a metaheuristic, MBO, and showed its convergence to the global optimum value. Adapting MBO, we then proposed a PI-inspired algorithm, HBPI, for infinite horizon discounted cost SDP problems, and its validity has been demonstrated by convergence analysis. The present paper focused on the theoretical properties of the algorithms. Of course, the future research will be an experimental evaluation on diverse dynamic stochastic optimization problems as they frequently occur in application contexts.

Convergence results are a first step to gain some confidence in the theoretical soundness of a heuristic algorithm. An important issue from a practical point of view are assertions on the speed of convergence to "good" solutions, depending on the chosen parameter values of the algorithm. Our results in this paper do not yet cover this issue, since the analysis is based on rather rough probabilistic bounds. But the convergence rate analysis of HBPI would be very difficult as there is no known result for PI. The purpose of this paper is to give a first demonstration of the capability of the MBO approach (in a theoretical perspective) for solving combinatorial optimization problems and stochastic sequential decision-making problems. More detailed investigations will be necessary to make the approach fruitful for applications.

Acknowledgements

The author would like to thank an anonymous reviewer for his/her many helpful comments and pointing out the previous errors for improving this paper. This work was supported by the 21st Century Frontier Program of the Ministry of Commerce, Industry and Energy in S. Korea.

References

1. Abbass, H.A. (2001), Marriage in honey bees optimization: a haplometrosis polygynous swarming approach. In: *Proc. of the IEEE Congress on Evolutionary Computation*, pp. 207–214.
2. Abbass, H.A. and Teo, J. (2001), A true annealing approach to the marriage in honey-bees optimization algorithm. In: *Proc. of the Inaugural Workshop on Artificial Life (AL'01)*, pp. 1–14.
3. Bentley, J. (1992), Fast algorithms for geometric traveling salesman problems, *ORSA Journal on Computing* 4(4), 387–411.
4. Bertsekas, D.P. (1995), *Dynamic Programming and Optimal Control, Volumes 1 and 2*. Athena Scientific.
5. Bertsekas, D.P. and Tsitsiklis, J.N. (1996), *Neuro Dynamic Programming*. Athena Scientific.
6. Bonabeau, E., Dorigo, M. and Theralaz, G. (1999), *Swarm Intelligence: From Natural to Artificial Systems*. Oxford Press.

7. Chang, H.S., Givan, R. and Chong, E.K.P. (2004), Parallel rollout for on-line solution of partially observable Markov decision processes, *Discrete Event Dynamic Systems: Theory and Application* 14(3), 309–341.
8. Chang, H.S., Gutjahr, W.J., Yang, J. and Park, S. (2004), An ant system based approach to Markov decision processes. In: *Proc. of the American Control Conference*, pp. 3820–3825.
9. Chang, H.S., Lee, H-G., Fu, M. and Marcus, S.I. (2005), Evolutionary policy iteration for solving Markov decision processes, *IEEE Trans. on Automatic Control* 50(11), 1804–1808.
10. Corne, D., Fl Glover and Dorigo, M. (eds.) (1999), *New Ideas in Optimization*. McGraw-Hill.
11. De Jong, K.A. (1975), *An analysis of the behavior of a class of genetic adaptive systems*. Ph.D. Thesis, Univ. of Michigan, Ann Arbor, MI.
12. Dietz, A. (1986), Bee genetics and breeding. In: Rinderer, T. (ed.), *Evolution*, Academic Press, pp. 3–22.
13. Dorigo, M. and Di Caro, G. (1999), The ant colony optimization metaheuristic. In: Corne, D. and Dorigo, M. (eds.), *New Ideas in Optimization*, McGraw-Hill, NY, USA, pp. 11–32.
14. Dorigo, M., Di Caro, G. and Stützle, T. (eds.) (2000), Special issue on “ant algorithms,” *Future Generation Computer Systems* 16(8).
15. Dorigo, M., Maniezzo, V. and Colorni, A. (1996), The Ant System: optimization by a colony of cooperating agents, *IEEE Trans. Systems Man Cybernet* 25, 29–41.
16. Eberhart, R.C. and Kennedy, J. (2001), *Swarm Intelligence*. Morgan Kaufmann.
17. Gutjahr, W. (2002), ACO algorithms with guaranteed convergence to the optimal solution, *Information Processing Letters* 82, 145–153.
18. Gutjahr, W. (2003), A generalized convergence result for the graph-based ant system metaheuristic, *Probability in the Engineering and Informational Sciences* 17, 545–569.
19. Johnson, D.S. and McGeoch, L.A. (1997), The traveling salesman problem: a case study in local optimization. In: Aarts, E.H.L. and Lenstra, J.K. (eds.), *Local Search in Combinatorial Optimization*, John-Wiley and Sons, Ltd., pp. 215–310.
20. Kaelbling, L.P., Littman, M. and Moore, A. (1996), Reinforcement learning: a survey, *Journal of Artificial Intelligence Research* 4, 237–285.
21. Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. (1983), Optimization by simulated annealing, *Science* 220, 45–54.
22. Littman, M., Dean, T. and Kaelbling, L. (1995), On the complexity of solving Markov decision problems. In: *Proc. 11th Annual Conf. on Uncertainty in Artificial Intelligence*, pp. 394–402.
23. Pardalos, P.M. and Resende, M. (eds.) (2002), *Handbook of Applied Optimization*. Oxford University Press.
24. Puterman, M.L. (1994), *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, New York.
25. Reeves, C.R. (1997), Genetic algorithms for the operations researcher, *INFORMS Journal on Computing* 9(3), 231–250.
26. Rudolph, G. (1994), Convergence analysis of canonical genetic algorithms, *IEEE Transactkions on Neural Networks* 5(1), 96–101.
27. Spears, W.M. and DeJong, K.A. (1991), An analysis of multipoint crossover. In: *Proc. 1990 Workshop of the Foundations of Genetic Algorithms*, pp. 301–315.
28. Srinivas, M. and Patnaik, L.M. (1994), Genetic algorithms: a survey, *IEEE Computer* 27(6), 17–26.
29. Sutton, R. and Barto, A. (2000), *Reinforcement Learning*. MIT Press.