International Journal of Automation and Computing 13(5), October 2016, 468-479
DOI: 10.1007/s11633-016-0997-z

Sample Bound Estimate Based Chance-constrained Immune Optimization and Its Applications

Zhu-Hong Zhang[1] Kai Yang[2] Da-Min Zhang[1]
[1]College of Big Data & Information Engineering, Guizhou University, Guizhou 550025, China
[2]College of Computer Science & Technology, Guizhou University, Guizhou 550025, China

Abstract: This work investigates a simple and practical bio-immune optimization approach to solve a kind of chance-constrained programming problem without known noisy attributes, after probing into a lower bound estimate of sample size for any random variable. Such approach mainly consists of sample allocation, evaluation, proliferation and mutation. The former two, depending on a lower bound estimate acquired, not only decide the sample size of random variable and the importance level of each evolving B cell, but also ensure that such B cell is evaluated with low computational cost; the third makes diverse B cells participate in evolution and suppresses the influence of noise; the last, which associates with the information on population diversity and fitness inheritance, creates diverse and high-affinity B cells. Under such approach, three similar immune algorithms are derived after selecting different mutation rules. The experiments, by comparison against two valuable genetic algorithms, have illustrated that these immune algorithms are competitive optimizers capable of effectively executing noisy compensation and searching for the desired optimal reliable solution.

Keywords: Chance-constrained programming, immune optimization, sample allocation, lower bound estimate, noise attenuation.

1 Introduction

Chance-constrained programming (CCP) or probabilistic programming is a kind of uncertain optimization which involves in constraints with noise. Despite of wide applications, it is studied scarcely because of being a challenging task. The main difficulty is that the noise environment influences seriously the optimized quality, execution efficiency and individual′s evaluation; hence, it is almost impossible to obtain the theoretical optimum of a given CCP problem. This makes the existing intelligent approaches become extremely difficult in solving a large number of practical optimization problems with noise. In the literature, some evolutionary computation-based achievements solving expected value optimization problems without any constraint restriction have been reported[1, 2]. Such work concentrates mainly on either analyzing performance characteristics of some existing evolutionary algorithms or extending them through designing special sampling schemes. However, these achievements become difficult when dealing with CCP problems, due to chance constraints or probabilistic inequalities. So, novel intelligent techniques are desired for such kind of problem.

In the early theoretical research on CCP, many researchers investigated how to handle CCP′s constraints under certain assumptions, e.g., convexity[3], linear approximation[4], transformation or stochastic simulation[5−7]. However, their methods might be difficult in solving practical CCP problems because of sophisticated transformation or computational complexity. Sahinidis[8] reviewed in detail the classical theory and methods developed to cope with uncertain programming problems, and also discussed multiple kinds of CCP approaches. Afterwards, several advanced methodologies, e.g., double-loop, single-loop and decoupled methods[9, 10], were suggested to solve general chance constraints. Deb et al.[10] proposed a hybrid optimization technique by integrating their multiobjective algorithm NSGA-II with the above methods. In their approach, an additional objective function is utilized to handle all the chance constraints, and hence the problem solving is reformulated as a bi-objective problem. Their method is desirable when only the optimized quality is considered. Up to now, some computational techniques for CCP problems have been reported[11−14]. For instance, Cao et al.[14] suggested two kinds of CCP models to evaluate the extra cost of uncertainty for the refinery short-term crude oil scheduling problem under uncertain demands of distillation units. The two models are converted respectively into the analytically equivalent ones, while the branch and bound method is used to solve them. All these techniques mentioned are restricted mainly to normally distributed random variables[10], while requiring that the CCP models be transformed into analytically deterministic programming ones[15, 16]. However, if so, they are extremely difficult when noise information is unknown[17]. Poojari and Varghese[18] developed a modified steady state genetic algorithm (SSGA) for general CCP, and proposed two kinds of genetic algorithms (i.e., SSGA-A and SSGA-B), by introducing two optimality scoring functions and specifying

Research Article
Manuscript received October 31, 2013; accepted July 15, 2015; published online July 25, 2016
This work was supported in part by National Natural Science Foundation of China (Nos. 61563009 and 61065010) and Doctoral Fund of Ministry of Education of China (No. 20125201110003).

a fixed sample size for each random variable. Our experiments have showed that the algorithms are useful when the noise is weak. Unfortunately, their fitness schemes deem easily those inferior individuals as better ones.

Since a number of authors demonstrated that bio-inspired immune algorithms were superior to several classical intelligent approaches for multimodal optimization problems[19−23], immune optimization has become increasingly popular. Several authors investigated how to utilize the existing immune algorithms to handle uncertain optimization problems[24−27]. Especially, de França et al.[27] proposed a modified artificial immune network by extending their original immune network opt-aiNet suitable for static optimization problems. Their principal work concentrates on some additional procedures to improve the overall performance of such immune network. To our knowledge, in addition to our work, immune algorithms for CCP problems have been rarely reported in the field of artificial immune systems.

In our previous work[28], an immune algorithm in noisy environments, coping with expected value optimization problems without constraints, was proposed based on the hypothesis test and simplified immune metaphors of the humoral immunity. The algorithm can suppress noises and rapidly discover the optimal solution of a given optimization problem, in terms of a dynamic suppression radius and a time-varying suppression probability. Further, after investigating an a priori estimate of sample size of the sample approximation problem related to CCP, we in this paper put forward a practical chance-constrained programming immune approach (CCPIA) to solve CCP problems without any known noise information, depending on the clonal selection theory in biological immunology. Under the approach, three similar immune algorithms are developed through selecting different mutation rules. The main purpose doing so is to examine CCPIA′s characteristics sufficiently. Subsequently, they are compared to SSGA-A and SSGA-B mentioned above, relying upon two theoretical test problems and three engineering problems. Experimental results have displayed their potential value.

2 Problem formulation

Consider the following CCP problem of the form (P_α):

$$\min\{\mathrm{E}[f(\boldsymbol{x},\xi)],\quad \boldsymbol{x}\in X_\alpha\}$$

where

$$X_\alpha=\{\boldsymbol{x}\in D: Pr\{G_i(\boldsymbol{x},\xi)\le 0\}\ge 1-\alpha_i, 1\le i\le I,\\ g_j(\boldsymbol{x})\le 0,\quad h_k(\boldsymbol{x})=0,\quad 1\le j\le J, 1\le k\le K\}.$$

In such model, $D\subset \mathbf{R}^p$ represents a bounded and closed domain; ξ is an unknown distributed random vector with support $\Xi\subset\mathbf{R}^d$; $f(\boldsymbol{x},\xi)$ stands for the stochastic objective function, and $G_i(\boldsymbol{x},\xi)$ is the i-th stochastic constraint; α_i is the i-th significance level; $g_j(\boldsymbol{x})$ and $h_k(\boldsymbol{x})$ are the deterministic constraints. Set $\alpha=(\alpha_1,\alpha_2,\cdots,\alpha_I)$. $\mathrm{E}[\cdot]$ and $Pr\{\cdot\}$ are the mathematical expectation and probability operators, respectively.

In order to deal with the above constraints, let $\Gamma(\boldsymbol{x})$ represent the total of constraint violations for a given $\boldsymbol{x}\in D$, i.e., the sum of violations for all the constraints. It satisfies that $\Gamma(\boldsymbol{x})=0$ if and only if $\boldsymbol{x}\in X_\alpha$; for $\boldsymbol{x},\boldsymbol{y}\in D$, if $\Gamma(\boldsymbol{x})<\Gamma(\boldsymbol{y})$, $\boldsymbol{x}$ is said to be better than $\boldsymbol{y}$. $\boldsymbol{x}\in X_\alpha$ is called a reliable solution, and an unreliable solution otherwise. $\boldsymbol{x}^*\in X_\alpha$ is called an optimal reliable solution only if $\mathrm{E}[f(\boldsymbol{x}^*,\xi)]\le \mathrm{E}[f(\boldsymbol{x},\xi)]$ for $\forall \boldsymbol{x}\in X_\alpha$. Usually, P_α is processed by solving a sample average approximation problem, due to the difficulty of calculating $\mathrm{E}[f(\boldsymbol{x},\xi)]$ and $Pr\{G_i(\boldsymbol{x},\xi)\le 0\}$ for a given $\boldsymbol{x}\in D$. Recently, there have been many sample approximation methods for chance constraints[29, 30]. Luedtke and Ahmed[29] acquired a lower bound estimate to the true optimal value for a CCP problem with only a joint probabilistic constraint, relying upon a sample approximation problem with a larger confidence level than the required confidence level. Their theoretical result is helpful in probing into a lower bound estimate of sample size for a Monte Carlo sample approximation problem of P_α. We now consider independent observations of the random vector ξ, i.e., $\widehat{\xi}^1,\widehat{\xi}^2,\cdots,\widehat{\xi}^M$ and $\xi^{1,i},\xi^{2,i},\cdots,\xi^{M,i}, 1\le i\le I$, where M is the sample size. The sample average approximation problem is usually defined as follows (P_M^β):

$$\min\{F_M(\boldsymbol{x})=\frac{1}{M}\sum_{l=1}^{M} f(\boldsymbol{x},\widehat{\xi}^l),\ \boldsymbol{x}\in X_M^\beta\}$$

where $\beta=(\beta_1,\beta_2,\cdots,\beta_I)$, $0<\beta_i<1, 1\le i\le I$, and

$$X_M^\beta=\{\boldsymbol{x}\in D:\frac{1}{M}\sum_{l=1}^{M}\Pi(G_i(\boldsymbol{x},\xi^{l,i})\le 0)\ge 1-\beta_i,\\ 1\le i\le I, g_j(\boldsymbol{x})\le 0,\ h_k(\boldsymbol{x})=0, 1\le j\le J, 1\le k\le K\}.$$

In the above approximation model, $\Pi(\cdot)$ is the indicator function which takes value 1 when $\cdot$ is true and 0 otherwise[29]; we say $\beta>\alpha$ if $\beta_i>\alpha_i$ with $1\le i\le I$. To solve P_α by means of P_M^β, we need to know the following estimate for which the proof can be found in Appendix.

Theorem 1. Let $\boldsymbol{x}_\alpha^*$ and $\widehat{\boldsymbol{x}}_M^\beta$ be the optima of P_α and P_M^β with the minima θ_α^* and $\widehat{\theta}_M^\beta$, respectively. If $\beta>\alpha$, and $a\le f(\boldsymbol{x},\xi)\le b$ for $(\boldsymbol{x},\xi)\in D\times\Xi$, then the following estimate holds for $\eta>0$,

$$Pr\{\theta_\alpha^*\ge\widehat{\theta}_M^\beta-\eta\}\ge 1-(c+2)\mathrm{e}^{\{-2\kappa^2M\}}\tag{1}$$

where $c>I$ and $\kappa=\min\left\{\dfrac{\eta}{b-a},\min\limits_{1\le i\le I}\dfrac{\beta_i-\alpha_i}{2\sqrt{\alpha_i}}\right\}$.

Remark 1. Usually, it is difficult to decide a and b in practice. In order to overcome this point, let η depend on $b-a$, e.g., $\eta=b-a$. This way, k is decided by $\min\{\frac{\beta_i-\alpha_i}{2\sqrt{\alpha_i}}, 1\le i\le I\}$, provided that the parameters, $\beta_i-\alpha_i, 1\le i\le I$, take small values.

Remark 2. A lower bound of M can be acquired with significance level δ, $0<\delta<1$, namely if

$$M\ge M_\delta\equiv\lfloor\frac{1}{2k^2}\log\frac{c+2}{\delta}\rfloor\tag{2}$$

we have

$$Pr\{\theta_{\alpha}^{*} \geq \widehat{\theta}_{M}^{\beta} - \eta\} \geq 1 - (c+2)\mathrm{e}^{-\log\frac{c+2}{\delta}} = 1 - \delta \quad (3)$$

where M_{δ} is dependent on α_i, β_i, c and δ, providing us a lower bound estimate of sample size of random vector ξ at a given candidate for CCPIA. Also, we can control M_{δ} between 100 and 250, provided that the settings of the parameters are rational; for example, when $\alpha = 0.05$, $\beta = 0.1$, $\delta = 0.1$ and $c = 1.01$, we acquire $M_{\delta} = 136$. More details can be found in Table 1 given in Section 5. Additionally, we say that a given candidate $\boldsymbol{x}$ in D with sample size M is empirically reliable if $\boldsymbol{x} \in X_{M}^{\beta}$ and unreliable otherwise.

3 Clonal selection theory

When an organism is exposed to an antigen, a second signal from T_h cells stimulates antigenic receptors of B cells to bind to such antigen. This stimulation causes that such B cells proliferate and differentiate into two different cell types (i.e., plasma and memory cells). If such plasma cells are active, they will secrete some antibodies to neutralize the triggering antigen. In addition, the memory cells will become long-lived ones. Once the specific antigen is found once again in the immune system, these memory cells will commence rapidly differentiating into plasma cells capable of producing high-affinity antibodies. Such theory involves in some important properties[31]:

1) Cell selection. Those B cells with high affinities to the invading antigen are chosen to change their pattern structures so that better B cells can be created.

2) Clonal expansion. Those stimulated B cells proliferate and differentiate into two different cell types. The plasma cells produce some clones with their clonal sizes proportional to their affinities. The memory ones will live in the immune system for a long time.

3) Hypermutation. During the clonal expansion, random changes are introduced in the variable region. Occasionally, one such change leads to an increase in the affinity of the lymphocytes. This process creates a variety of new B cells, where the mutation probability of a B-cell is inversely proportional to its affinity to the antigen. After so, some worse clonal cells will encounter suppression.

The clonal selection theory formulates an evolutionary process, which may be simulated to design CCPIA for CCP problems. Theoretically, after an antigen enters the immune system, high-affinity B cells are selected to reproduce some clones, while these clones have opportunities to change their genes according to their mutation probabilities. Then, all mutated clones and their parents maintain the relation of suppression and stimulation through immune regulation. Additionally, some new cells from the bone marrow are created to strike a dynamical balance.

4 Immune optimization approach

4.1 Algorithm formulation

For simplicity, we only cite some main immune properties in the clonal selection principle mentioned above. Corresponding to the approximation model P_{M}^{β} as in Section 2, a real-encoded B-cell is viewed as a candidate solution, while the antigen is regarded as the problem itself. Our goal is to acquire the optimal reliable solution of P_{α} by solving P_{M}^{β} with gradually increasing M. In the following algorithm design, we require that the sample size of random vector ξ be decided by the importance of B-cell in a given population, namely different B cells are attached different sample sizes. Thereby, the sample size M in P_{M}^{β} for CCPIA is replaced by $m(\boldsymbol{x})$ defined below with $m(\boldsymbol{x}) \leq M_{\delta}$, due to computational complexity. The following pseudo-procedure formulates our immune optimization approach:

```
CCPIA pseudo-procedure
1) Input: $N$-population size, $M_{\delta}$-lower bound, $m$-initial sample size, $M_c$-clonal sample size and $T$-maximal iteration
2) Set $n = 1$
3) Initialization, $A = \text{Population}(N, m)$
4) While $n \leq T$ do
5)     Execute sample allocation: $\text{Allocation}(A, M_{\delta}, n)$
6)     Perform evaluation: $\text{Evaluation}(A)$
7)     Enforce reproduction, $B = \text{Proliferation}(A)$
8)     Implement mutation, $C = \text{Mutation}(B, M_c)$
9)     Carry out population update, $A = \text{Update}(A \cup C, N)$
10)    $n = n + 1$
11) End while
12) Output: The best B-cell.
```

The above pseudo-procedure presents an optimization mechanism for P_{M}^{β}. Step 3 generates an initial population with size N, which requires to calculate the empirical objective value and the sum of empirical constraint violations for each cell; steps 5 to 10 are a run period, which ultimately creates the desired solution of P_{α}. The modules are designed below.

Theoretically, if the sample size of ξ is large enough, we can obtain the approximate solution of P_{α} by solving P_{M}^{β} with a higher confidence vector than the required confidence vector α, but the computational cost is expensive. Thus, when executing the above CCPIA, we ask that the sample size of ξ at each B-cell increases gradually. Precisely, such cell is with a dynamically increasing sample size, which helps CCPIA reduce the total of evaluations within a single run. To this end, let M_n, given in this work by $\lfloor mN\sqrt{n+1} \rfloor$, represent the sum of sample sizes for B cells in the current population at the moment n. Thereafter, M_n is allocated to B cells in the population. So, when gradually increasing the sample sizes of evolving B cells, CCPIA searches for the approximate optimum of P_{α}. Further, Theorem 1 gives us a lower bound (M_{δ}) utilized to limit the increasing sample size at each B cell. This helps us design a sample allocation scheme below.

In $\text{Allocation}(A, M_{\delta}, n)$, all B cells in A are first divided into two subpopulations of empirically reliable solution population X_1 and empirically unreliable solution population X_2; second, let N_1 denote the size of X_1. So, the sample size of ξ for a B cell $\boldsymbol{x}$ in A is defined by

$$m(\boldsymbol{x})=\begin{cases}\min\{M_\delta,\frac{M_n}{N_1}\}, & \boldsymbol{x}\in X_1\\ \min\{M_\delta,\frac{M_n}{N}\}, & \boldsymbol{x}\in X_2.\end{cases}\tag{4}$$

Here, the main motivation doing so is to guarantee that any empirically reliable solution can get a larger sample size than any empirically unreliable one. Also, considering computational cost, the cells in X_1 or X_2 are required to attach the same sample size.

Evaluation(A) decides the importance of each cell in A. In other words, based on the above sample allocation scheme, we calculate and normalize the empirical average objective values and constraint violations for B cells in A, and correspondingly obtain $\widehat{\mu}(\boldsymbol{x})$ and $\widehat{\Gamma}(\boldsymbol{x})$ with $\boldsymbol{x}\in A$, where $\widehat{\mu}(\boldsymbol{x})\in[-1,1]$ and $\widehat{\Gamma}(\boldsymbol{x})\in(0,1]$. This way, the importance of a B-cell $\boldsymbol{x}$ in A can be measured through its affinity $Aff(\boldsymbol{x})$ given by

$$Aff(\boldsymbol{x})=\begin{cases}\dfrac{N-l(\boldsymbol{x})}{N}(2-\widehat{\mu}(\boldsymbol{x})), & \boldsymbol{x}\in X_1\\ \dfrac{N-N_1-m(\boldsymbol{x})}{N}(1-\widehat{\Gamma}(\boldsymbol{x})), & \boldsymbol{x}\in X_2\end{cases}\tag{5}$$

where $l(\boldsymbol{x})$ and $m(\boldsymbol{y})$ represent the importance levels of $\boldsymbol{x}$ in X_1 and $\boldsymbol{y}$ in X_2, respectively; for example, after increasingly ranking all elements in X_1 according to their empirical objective values, e.g., $\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots$, we get that $l(\boldsymbol{x}_2)=2$. Equation (5) illustrates that $Aff(\boldsymbol{x})>Aff(\boldsymbol{y})$ if $\boldsymbol{x}\in X_1$ and $\boldsymbol{y}\in X_2$, because of $l(\boldsymbol{x})<N_1+m(\boldsymbol{y})$ and $2-\widehat{\mu}(\boldsymbol{x})\geq 1$.

Proliferation(A) keeps diverse cells in A and decides their clonal sizes attached. Precisely, we first rank decreasingly all cells in A in terms of their affinities decided by (5), e.g., $\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_N$. Second, introduce a niche-like suppression radius $\delta(\boldsymbol{x})$ to eliminate redundant cells in A,

$$\delta(\boldsymbol{x})=\begin{cases}\dfrac{Aff(\boldsymbol{x})^\nu}{\sigma_A+Aff(\boldsymbol{x})^\nu}, & f(\boldsymbol{x},\xi)\equiv f(\boldsymbol{x})\\ \dfrac{(\sigma_A)^\nu}{(\sigma_A)^\nu+\widehat{\sigma}(\boldsymbol{x})}, & \text{otherwise}\end{cases}\tag{6}$$

with $\nu\geq 1$, where σ_A denotes the variance of empirical average objective values for all B cells in A; $\widehat{\sigma}(\boldsymbol{x})$ represents the variance of observations for $\boldsymbol{x}$ with sample size $m(\boldsymbol{x})$. Equation (6) indicates an important fact in the case where $f(\boldsymbol{x},\xi)$ depends on ξ, namely if σ_A is larger, there are many B cells in A to be eliminated. So, we divide A into l disconnected subclasses, $\mathrm{O}(\boldsymbol{x}_{m_i}), 0\leq i<l$, where $m_0=1$, and $\mathrm{O}(\boldsymbol{x}_{m_i})$ consists of elements in A whose affinities are between $aff(\boldsymbol{x}_{m_i})-\delta(\boldsymbol{x}_{m_i})$ and $aff(\boldsymbol{x}_{m_i})$. Further, $\boldsymbol{x}_{m_i}$ as a surviving cell reproduces a clonal subpopulation $Cl(\boldsymbol{x}_{m_i})$ whose size equals that of $\mathrm{O}(\boldsymbol{x}_{m_i})$. Thereafter, those surviving cells in A are kept with their clonal sizes. Such module can guarantee that diverse B cells are admitted to survive and that those B cells with higher affinities can propagate many more clones.

Mutation(B, M_c) creates N clones mutated. Each clone $\boldsymbol{x}$ in B mutates its genes through a special mutation rule, e.g., Gaussian or nonuniform mutation, in which its mutation probability is defined by

$$p_m(\boldsymbol{x})=\frac{(\sigma_B)^\nu}{(\sigma_B)^\nu+aff(\boldsymbol{x})}.\tag{7}$$

Further, after being assigned the same small sample size M_c, all mutated clones are required to calculate their empirical average objective values and constraint violations. In particular, if $f(\boldsymbol{x},\xi)$ depends on ξ, we update the empirical values of the mutated clones by inheriting those of their parents. Namely, for a given mutated clone $\boldsymbol{x}'$ with its parent $\boldsymbol{x}$, if $\widehat{\mu}(\boldsymbol{x}')$ is between $\widehat{\mu}(\boldsymbol{x})-\widehat{\sigma}(\boldsymbol{x})$ and $\widehat{\mu}(\boldsymbol{x})+\widehat{\sigma}(\boldsymbol{x})$, then $\widehat{\mu}(\boldsymbol{x}')$ and $\widehat{\Gamma}(\boldsymbol{x}')$ are updated by $\frac{\widehat{\mu}(\boldsymbol{x})+\widehat{\mu}(\boldsymbol{x}')}{2}$ and $\frac{\widehat{\Gamma}(\boldsymbol{x})+\widehat{\Gamma}(\boldsymbol{x}')}{2}$ respectively. This is because we demand that M_c be small as possible, which is helpful in reducing the computational cost of such scheme and also in suppressing the noise influence on the optimized quality.

Update($A\cup C, N$) is composed of those mutated clones in C and some parents in A without any empirical constraint violation. If the number of these cells is beyond N, the better elements are stored; otherwise, pick up some remaining better parents so that a new population with size N is formed.

4.2 Computational complexity

In a run period, CCPIA only needs to calculate the empirical values of B cells appearing in Steps 6 and 8. So, we can estimate its computational cost in the worst case given by the following conclusion. The proof is given in the Appendix.

Theorem 2. If $M_c<M_\delta$, the computational complexity of CCPIA in the worst case is $\mathrm{O}(M_\delta N)$.

As associated to the proof of the above theorem, CCPIA needs at most $\widehat{m}_c$ times to compute within a run period with $\widehat{m}_c\leq N((M_\delta+M_c+1)I+2(J+K))$, where I, J and K are the numbers of constraints mentioned in Section 2. We know that either SSGA-A or SSGA-B mentioned in Section 1, which is selected to participate in comparison against CCPIA, needs to calculate m_s times within an iterative period with $m_s=N(\overline{M}I+J+K)$, where $\overline{M}$ denotes the same sample size attached with each individual. In the corresponding literature, the authors took $\overline{M}$ as 300, due to the optimized quality. We can assert that the computational cost of CCPIA is lower than that of each of the two approaches, as $M_\delta+M_c$ is required to be not beyond 260. For example, if taking $N=80$, $M_c=10$, $I=J=K=1$, $\alpha=0.025$, $\beta=0.075$, $\delta=0.01$, $c=1.5$ and $\overline{M}=300$, (2) derives $M_\delta=117$, and hence $\widehat{m}_c\leq 10\,560$, but $m_s=24\,160$.

5 Numerical experiments

Our experiments are executed on a personal computer with CPU/3.30 GHz and RAM/2.98 MB. Based on CCPIA, three similar immune algorithms, CCPIA-A, CCPIA-G and CCPIA-U, are derived through designating the mutation rule in the mutation scheme in order as polynomial mutation[32], classical Gaussian mutation and nonuniform mutation. These three algorithms are used to examine CCPIA′s characteristics including its convergence, solution quality and efficiency. As we mention in introduction, SSGA-A and SSGA-B[18] are two competitive approaches for general single-objective CCP problems. They are chosen to participate in comparison, depending on the following five representative CCP examples. All these five algorithms are required to execute respectively 100 single runs on the approximation model of each test example. Let their population sizes N be 40, and the maximal iteration number 200. Especially, other parameter values of SSGA-A and SSGA-B are taken from the reported literature[18], i.e., $p_{cross} = 0.7$, $p_{mut} = 0.1$, $p_{rept} = 0.5$, $\lambda_{\min} = 0.01$, $\lambda_{\max} = 0.99$, and sample size 300 for each individual. After experimental trials for CCPIA, we take $m = 20, M_c = 10$ and $\nu = 1.2$, while the settings of parameters, presented in (2) in Section 2 are given in Table 1 below. Also, the reason why we select the following examples with Gaussian distribution is because some theoretical maxima or minima are known, which helps us analyze the statistical results. Especially, it is pointed out that, whereas Examples 1 and 2 below are two stochastic linear programming problems, their equivalent models are nonlinear. Also, in order to examine the characteristics of the above approaches, our work is to directly solve their sample approximation problems, instead of their analytically equivalent models.

Example 1. Feed mixer problem[18]

Let x_1, x_2, x_3 and x_4 represent the proportion of barley, oats, sesame flakes and groundnut meal in the mix, respectively; $\xi = (\xi_1, \xi_2, \xi_3, \xi_4)$ is a random vector with its elements denoted respectively by the amount of protein for such four materials. The feed mixer model is described as a stochastic programming model P_α

$$\text{Max } 24.55x_1 + 26.75x_2 + 39.00x_3 + 40.50x_4$$
$$\text{s.t.,} \begin{cases} 2.3x_1 - 5.6x_2 - 11.1x_3 - 1.3x_4 \geq 5 \\ Pr\{x_1\xi_1 + x_2\xi_2 + x_3\xi_3 + x_4\xi_4 \geq 21\} \geq 1 - \alpha \\ x_1 + x_2 + x_3 + x_4 = 1, x_1, x_2, x_3, x_4 \geq 0 \\ \xi_1 \sim N(12, 0.53), \xi_2 \sim N(11.9, 0.44) \\ xi_3 \sim N(41.8, 4.5), \xi_4 \sim N(52.1, 0.79). \end{cases}$$

In this experiment, we take $\alpha = 0.05$ and 0.2, and obtain the same theoretical maximum 39.933 7. After each algorithm executes 100 times directly on the approximation model with $\beta = 0.1$ or 0.3, those solutions found are required to be evaluated 10^5 times so as to acquire their theoretical constraint violations. The statistical results are listed in Table 2, while Fig. 1 below displays the average search curves.

Table 2 presents that SSGA-A cannot find any empirically reliable solution when $\beta = 0.1$ or 0.3, due to $SR = 0$. SSGA-B is of better performance than SSGA-A, because of the values on CR, SR and RER below. On the other hand, we also note that other three approaches, CCPIA-A, CCPIA-G and CCPIA-U, can all acquire empirically reliable solutions in each execution, owing to their values on CR and SR. The values on $Std.Dev$ and CI below illustrate that all the algorithms can all gain stable searching effects, as their confidence intervals found are narrow; thus, these algorithms should be locally or globally convergent (see Fig. 1). Further, through the values on $Mean$ and RER below, we know that the immune approaches can all obtain their approximate optimal solutions in the case of α =0.05 or 0.2, as the error rates on RER, caused by them are at most 1.5%, namely their average values found are close to the theoretical maximum. Relatively, CCPIA-A achieves the best search performance. These indicate that different mutation rules make CCPIA acquire different optimized qualities, while the sample allocation scheme helps CCPIA accelerate to seek the desired reliable solution and suppress the noise influence as well. We also notice that the five approaches can all obtain respectively similar results in the cases of $\beta = 0.1$ and 0.3, which is because those reliable solutions might belong to the same region when α takes 0.05 and 0.2. Further, through the values on AT, we see that each of the five algorithms presents the same efficiency when β takes 0.1 and 0.3. We also observe that SSGA-A has the highest performance efficiency, but it can only find unreliable solutions with large constraint violations. In addition, CCPIA-A has a higher efficiency than each of SSGA-B, CCPIA-G and CCPIA-U; CCPIA-G and CCPIA-U are secondary. Fig. 1 hints that SSGA-A gets early into local search; CCPIA-U can converge rapidly, but CCPIA-G is convergent slowly. Additionally, it seems to be true that SSGA-A can achieve the best effect through such figure, because of its mean value ($Mean$) under $\beta = 0.1$ or 0.3. In fact, it can only acquire the worst effect, as it cause large constraint violations. This phenomenon will also appear in the following experiments.

Example 2. Kilosa farmer problem[18]

Poojari and Varghese[18] introduced the yield problem of crops for a family in Kilosa. Such problem was solved

Table 1 Settings of parameters as in (2) for different examples

Example	Case 1					Case 2				
	α	β	δ	c	M_δ	α	β	δ	c	M_δ
1	0.05	0.1	0.1	1.01	136	0.2	0.3	0.1	1.01	136
2	0.6	0.7	0.9	2.01	179	0.8	0.9	0.85	2.001	248
3	0.05	0.1	0.1	2.01	147	0.2	0.27	0.5	2.01	167
4	0.1	0.17	0.1	9.01	191	0.2	0.27	0.8	9.01	214
5	0.9	0.97	0.9	0.01	295	0.8	0.87	0.9	0.01	262

Table 2 Comparison of statistical values found for Example 1 with 100 runs per algorithm

	Opt.	β	*Mean*	*Std.Dev*	*CI*	*SR* (%)	*CR*	*RER* (%)	*AT*
SSGA-A			40.496	0.002	[40.496, 40.497]	0	3.70	2.9	0.28
SSGA-B			39.927	0.010	[39.925, 39.928]	98	10^{-7}	1.5	1.67
CCPIA-A	39.9337	0.10	39.933 4	10^{-4}	[39.933 4,39.933 5]	100	0	1.5	0.66
CCPIA-G			39.882	0.030	[39.877, 39.887]	100	0	1.4	0.77
CCPIA-U			39.932 8	0.001	[39.932 7, 39.932 9]	100	0	1.5	0.74
SSGA-A			40.497	0.002	[40.496, 40.497]	0	3.70	2.9	0.28
SSGA-B			39.926	0.011	[39.925, 39.927]	96	10^{-7}	1.5	1.68
CCPIA-A	39.9337	0.30	39.933 4	10^{-4}	[39.933 4, 39.933 5]	100	0	1.5	0.66
CCPIA-G			39.879	0.041	[39.875, 39.883]	100	0	1.4	0.77
CCPIA-U			39.932 9	0.001	[39.932 8, 39.932 9]	100	0	1.5	0.74

CI: Confidence interval, *SR*: Rate of empirically reliable solutions found among all solutions acquired, *CR*: Mean of constraint violations of solutions found, *RER*: Relative error rate between the theoretical maximum (or minimum) and the empirical mean (*Mean*), *AT*: Average time (second).

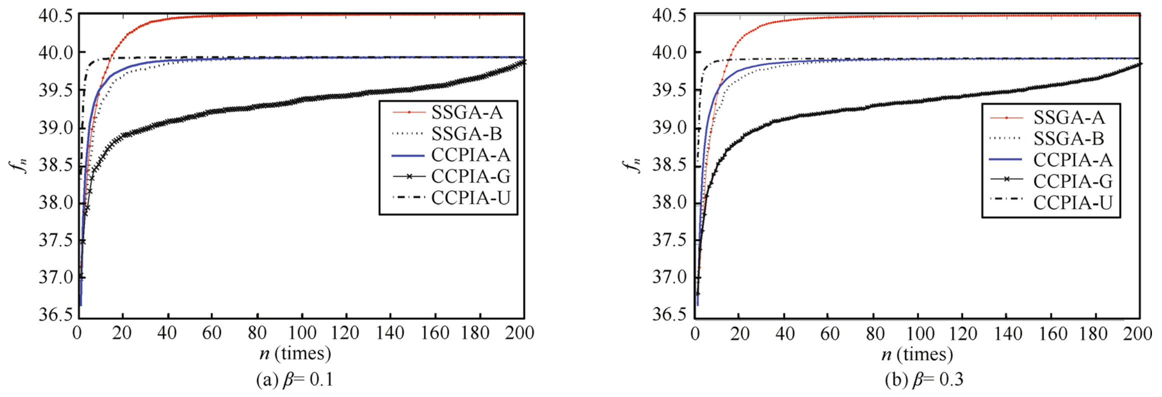

Fig. 1 Comparison of average search curves with 100 runs for Example 1. f_n is the average of the 100 best objective values at the n-th iteration with 100 executions.

through developing a stochastic programming model. Namely, let x_1 and x_2 be the respective acreage of maize and sorghum per hectare; ε_1 and ε_2 are their random yields per hectare with normal distributions; ξ is the total random rainfall during the growing season. The model is formulated below:

$$\text{Min}\ \ x_1 + x_2$$
$$s.t., \begin{cases} Pr\{2.8\xi_1 x_1 + 2.8\xi_2 x_2 \geq 44\} \geq 1-\alpha \\ Pr\{6.4\xi_1 x_1 + 8.0\xi_2 x_2 \geq 89\} \geq 1-\alpha \\ 0 \leq x_i \leq 20, i=1,2, \xi_1 = 0.02\zeta - 1.65 + \varepsilon_1 \\ \xi_2 = 0.008\zeta + 5.92 + \varepsilon_2 \\ \zeta \sim N(515.5, \sqrt{18\,769}), \varepsilon_1, \varepsilon_2 \sim N(0,10). \end{cases}$$

This problem is more difficult than Example 1 because of strong noises with large variances. The main purpose choosing it in this work is to examine whether the above approaches can effectively suppress the noise influence on the optimized quality. In this experiment, we take $\alpha = 0.6$ and 0.8, and hence obtain the theoretical minima 1.383 8 and 0.904 7 for the equivalent analytical model, respectively. Further, similarly dealing with Example 1, we acquire Table 3 below by directly solving P_M^β. The average search curves are given in Fig. 2. Table 3 reveals some characteristics of the above approaches. In the case of $\beta = 0.7$, the values on *CR* and *SR* hint that SSGA-A and SSGA-B fail to solve such P_M^β, as they can only find some empirically unreliable solutions with somewhat large constraint violations and relative error rates. The main reason is because their individual evaluation schemes deem easily inferior solutions as better ones during a single run. However, the immune approaches can all find many empirically reliable solutions for multiple single runs; in particular, despite the somewhat large values on *Mean* and *Std.Dev* in Table 3, CCPIA-A can obtain the best effect because of the values on *SR* and *RER*. Its empirical mean found is close to the theoretical minimum (1.383 8 or 0.904 7), while such theoretical value is next to its confidence interval, e.g., [0.999 8, 1.015 2].

The statistical values, obtained by the three immune approaches illustrate further that different mutation rules make CCPIA present different performance characteristics. In the case of $\beta = 0.9$, the similar characteristics of the immune approaches can be known from Table 3; CCPIA-A and CCPIA-U can almost find empirically reliable solutions during each run, while their average values are close to the theoretical minimum by comparison against the results in the case of $\beta = 0.7$. Besides, Fig. 2 shows that SSGA-A and SSGA-B get into local search but the immune approaches are convergent. Additionally, Table 3 also presents that SSGA-B spends the most runtime to execute a single run; CCPIA-A is secondary. Relatively, SSGA-A needs the least time to complete the process of optimization, but its performance effect is worst.

Example 3. Non-convex problem[18]

$$\text{Min} \quad -9x_1^2+10x_1x_2-50x_1+8x_2+460$$

$$s.t., \begin{cases} Pr\{x_1-0.277x_2^2+0.235x_2\leq 3.718+\xi_1\}\geq 1-\alpha \\ Pr\{x_1+0.019x_2^3-0.446x_2^2+3.98x_2\leq \\ \quad 15.854+\xi_2\}\geq 1-\alpha \\ 0\leq x_i\leq 4, i=1,2,\eta\sim N(0,1),\xi_1\sim N(0,0.01) \\ \xi_2\sim N(-5,0.02). \end{cases}$$

This is an extended version of the theoretical non-convex analytical optimization problem. When α takes 0.05 and 0.20, we acquire the theoretical minima of 151.604 and 150.673, respectively. Similar to the above experiments, the statistical results are listed in Table 4. The average search curves are drawn in Fig. 3. Through Table 4, the values on SR, CR and RER hint that when $\beta=0.1$, CCPIA-P and CCPIA-U can all find some empirically reliable solutions with small relative error rates (RER) for many runs. Moreover, their average values can all approach the theoretical minimum. Relatively, CCPIA-U is best (see Fig. 3). Although CCPIA-G can also acquire the desired value close to the theoretical minimum, it can hardly find empirically reliable solutions in each run. Obviously, SSGA-A and SSGA-B cannot gain their desired effects. In the case of $\beta=0.27$, the similar conclusion can be found, in which the major difference is that SSGA-B can obtain some empirically reliable

Table 3 Comparison of statistical values found for Example 2 with 100 runs per algorithm

	Opt.	β	*Mean*	*Std.Dev*	*CI*	*SR* (%)	*CR*	*RER* (%)	*AT*
SSGA-A			0.000 17	10^{-4}	[0.000 16, 0.000 17]	0	0.6	99.9	0.16
SSGA-B			0.009	0.08	[0.005 4, 0.011 8]	0	0.59	99	3.49
CCPIA-A	1.383 8	0.7	1.008	0.20	[0.999 8, 1.015 2]	24	0.088	27.2	1.87
CCPIA-G			0.8497	0.16	[0.8436 7, 0.855 7]	13	0.14	38.6	1.72
CCPIA-U			0.9532	0.21	[0.945 3, 0.961 0]	45	0.092	31.1	1.52
SSGA-A			0.0002	10^{-4}	[0.000 2, 0.000 22]	0	0.2	16	1
SSGA-B			0.0002	10^{-4}	[0.000 2, 0.000 21]	0	0.2	99.9	3.49
CCPIA-A	0.909 4	0.9	0.7728	0.060	[0.772 1, 0.773 5]	91	10^{-4}	15	2.14
CCPIA-G			0.7131	0.028	[0.712 7, 0.713 4]	59	0.004	21.6	1.98
CCPIA-U			0.7409	0.047	[0.740 4, 0.741 5]	87	0.0038	18.5	1.78

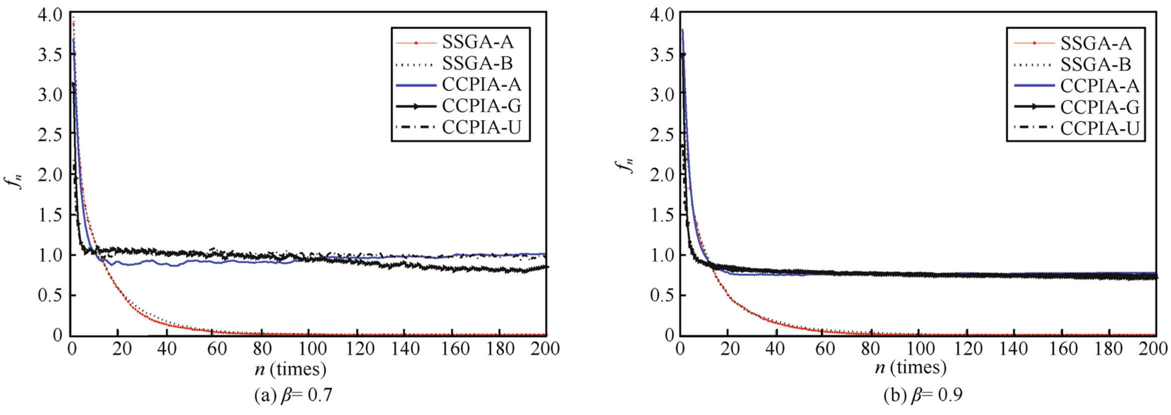

Fig. 2 Comparison of average search curves with 100 runs for Example 2

Table 4 Comparison of statistical values found for Example 3

	Opt.	β	*Mean*	*Std.Dev*	*CI*	*SR* (%)	*CR*	*RER* (%)	*AT*
SSGA-A			116.013	0.011	[116.011, 116.015]	0	0.9	23.5	0.07
SSGA-B			116.375	3.615	[115.781, 116.969]	1	0.891	23.2	1.37
CCPIA-A	151.604	0.1	153.763	11.363	[151.898, 155.629]	10	0.304	1.4	0.64
CCPIA-G			149.795	0.541	[149.706, 149.884]	2	0.500	1.2	0.57
CCPIA-U			151.394	0.484	[151.315, 151.474]	83	0.024	0.1	0.59
SSGA-A			116.012	0.008	[116.011, 116.013]	0	0.73	23	0.07
SSGA-B			117.052	5.942	[116.402, 117.703]	3	0.708	22.3	1.38
CCPIA-A	150.673	0.27	150.555	5.541	[149.948, 151.161]	19	0.233	0.1	0.69
CCPIA-G			149.665	0.465	[149.614, 149.716]	2	0.331	0.7	0.59
CCPIA-U			150.661	0.310	[150.627, 150.695]	89	0.018	0.01	0.62

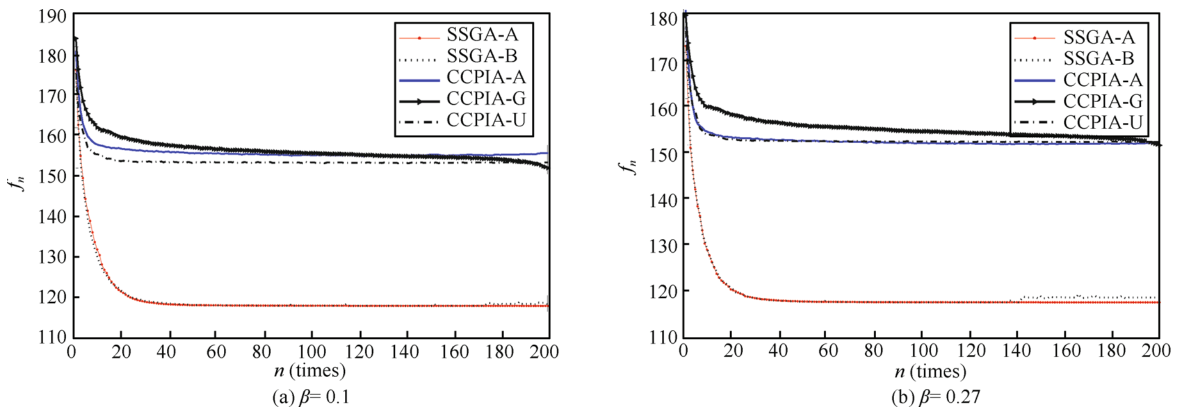

Fig. 3 Comparison of average search curves with 100 runs for Example 3

solutions during some runs. Further, the statistical values on AT show that SSGA-A has the highest efficiency but the worst effect; SSGA-B has the lowest efficiency; CCPIA-U has a lower efficiency and the best effect.

Example 4. Car side-impact problem[10]

The car side-impact problem is described by a stochastic programming model. It includes 7 decision variables, i.e., $(x_1, x_2, \cdots, x_7)$, 4 random variables $(\xi_1, \cdots, \xi_4)$ and 9 stochastic constraints. The mathematical model is given as follows:

$$\text{Min}\quad 1.98 + 4.9x_1 + 6.67x_2 + 6.98x_3 + 4.01x_4 + 1.78x_5 + 10^{-5}x_6 + 2.73x_7,$$

$$s.t.\ \begin{cases} 1.16 - 0.371\,7x_2x_4 - 0.009\,31x_2\xi_3 - 0.484x_3\xi_2 + 0.01343x_6\xi_3 \le 1 \\ 0.261 - 0.015\,9x_1x_2 - 0.188x_1\xi_1 - 0.019x_2x_7 + 0.014\,4x_3x_5 + 0.875\,7x_5\xi_3 + \\ \quad 0.080\,45x_6\xi_2 + 0.001\,39\xi_1\xi_4 + 0.000\,015\,75\xi_3\xi_4 \le 0.32 \\ 0.214 + 0.008\,17x_5 - 0.131x_1\xi_1 - 0.070\,4x_1\xi_2 + 0.030\,99x_2x_6 - 0.018x_2x_7 + \\ \quad 0.020\,8x_2\xi_1 + 0.121x_3\xi_2 - 0.003\,64x_5x_6 + 0.000\,771\,5x_5\xi_3 - 0.000\,535\,4x_6\xi_3 + \\ \quad 0.001\,21\xi_1\xi_4 + 0.001\,84\xi_2\xi_3 - 0.018x_2^2 \le 0.32 \\ 0.74 - 0.61x_2 - 0.163x_3\xi_1 + 0.001\,232x_3\xi_3 - 0.166x_7\xi_2 + 0.227x_2^2 \le 0.32 \\ 28.98 + 3.818x_3 - 4.2x_1x_2 + 0.020\,7x_5\xi_3 + 6.63x_6\xi_2 - 7.77x_7\xi_1 + 0.32\xi_2\xi_3 \le 32 \\ 33.86 + 2.95x_3 + 0.179\,2\xi_3 - 5.057x_1x_2 - 11x_2\xi_1 - 0.021\,5x_5\xi_3 - 9.98x_7\xi_1 + \\ \quad 22\xi_1\xi_2 \le 32, 46.36 - 9.9x_2 - 12.9x_1\xi_1 + 0.110\,7x_3\xi_3 \le 32, \\ 4.72 - 0.5x_4 - 0.19x_2x_3 - 0.012\,2x_4\xi_3 + 0.009\,325x_6\xi_3 + 0.000\,191\xi_4^2 \le 4 \\ 10.58 - 0.674x_1x_2 - 1.95x_2\xi_1 + 0.020\,54x_3\xi_3 - 0.019\,8x_4\xi_3 + 0.028x_6\xi_3 \le 9.9 \\ 16.45 - 0.489x_3x_7 - 0.843x_5x_6 + 0.043\,2\xi_2\xi_3 - 0.055\,6\xi_2\xi_4 - 0.000\,786\xi_4^2 \le 15.7 \\ \eta \sim N(0,1), \xi_1 \sim N(0.35, 0.006), \xi_2 \sim N(0.192, 0.006), \xi_3, \xi_4 \sim N(0, 10), \\ x_1, x_3, x_4 \in [0.5, 1.5], x_2 \in [0.45, 1.35], x_5 \in [0.875,\ 2.65], x_6, x_7 \in [0.4,\ 1.2]. \end{cases}$$

We transform the stochastic constraints into chance constraints with the same confidence level $1-\alpha$. In this experiment, we take α =0.1 and 0.2, and obtain the best objective values 15.57 and 15.576, respectively. Similar to the above experiments, the statistical results are listed in Table 5. Obviously, the results in Table 5 below illustrate that solving the above problem is difficult because of multiple constraints and strong noises. We see that SSGA-A and SSGA-B fail to solve it when β =0.17 or 0.27, due to their large constraint violations. However, the three immune approaches, in particular CCPIA-A and CCPIA-U, are all suitable for such problem. They can not only find empirically reliable solutions during each run, but also only cause small relative error rates and variances; CCPIA-U can acquire the best effect. The average values, obtained by them are almost or completely equal to the best value in the case of β =0.17 or 0.27. Whereas SSGA-A is locally convergent with the highest efficiency, it causes large constraint violations for many single runs. On the other hand, the three immune approaches can rapidly find the desired solutions, which demonstrates further that the operators in CCPIA are rational and useful.

Example 5. Multimodal CCP problem

$$\text{Max}\quad \sum_{i=1}^{100} \sin(\pi i x_i) + \eta$$

$$\text{s.t.,} \begin{cases} Pr\{\sum\limits_{i=1}^{100} \zeta_i x_i \le 500\} \ge 1-\alpha \\ Pr\{\sum\limits_{i=1}^{100} \eta_i x_i \le 7000\} \ge 1-\alpha \\ 0 \le x_i \le 10, \eta \sim N(0,1), \quad \zeta_i \sim \exp(1.2), \\ \eta_i \sim \log N(0.8, 0.6). \end{cases}$$

Table 5 Comparison of statistical values found for Example 4

	Best	β	*Mean*	*Std.Dev*	*CI*	*SR* (%)	*CR*	*RER* (%)	*AT*
SSGA-A			15.581	0.002	[15.581, 15.581 6]	0	85 904	0.01	0.776
SSGA-B			22.537	0.364	[22.487, 22.586]	0	858 99	44.7	17.40
CCPIA-P	15.57	0.17	15.576 7	0.0003	[15.576 7, 15.576 7]	100	0	0.001	6.06
CCPIA-G			15.712	0.092	[15.700, 15.725]	100	0	0.8	6.06
CCPIA-U			15.576	10^{-4}	[15.576, 15.576]	100	0	10^{-4}	6.05
SSGA-A			15.581	0.003	[15.581, 15.581 7]	0	85903	0.03	0.78
SSGA-B			22.623	0.381	[22.581, 22.665]	0	85898	45.2	17.41
CCPIA-P	15.576	0.27	15.576 7	0.0004	[15.576 7, 15.576 8]	100	0	10^{-3}	6.06
CCPIA-G			15.723	0.121	[15.71, 15.736]	100	0	1	6.07
CCPIA-U			15.576	10^{-4}	[15.576, 15.576]	100	0	10^{-4}	6.06

This is an extended version of one three-dimensional chance-constrained programming problem[33], including 100 variables and 3 stochastic variables. Solving such problem becomes extremely difficult because of non-normal distribution, multi-modality and high dimensionality. In this experiment, since the constraints are crucial, a large significance level is defined, i.e., $\alpha = 0.8$ or 0.9. Relying upon the parameter settings in Table 1, we obtain two best reliable solutions with the best objective values 363.925 and 382.345 when $\alpha = 0.9$ and 0.8, respectively. After the five approaches execute 100 times respectively, their respective statistical results are listed in Table 6.

Through Table 6, the five approaches can acquire stable results when $\alpha = 0.8$ and 0.9, in other words, they can present stable search performance. The values on CR indicate that SSGA-A and SSGA-B can not find reliable solutions for some runs during 100 executions, and especially SSGA-B results in large constraint violations. This hints that the same constraint-handling scheme in both SSGA-A and SSGA-B needs to make further improvements. However, CCPIA-A and CCPIA-G can always obtain reliable solutions for each execution, and meanwhile CCPIA-U causes constraint violation only for a few executions. This exhibits that the latter three approaches behave well with some merits of effective constraint handling and strong population exploration.

Since some of the solutions, found by SSGA-A and SSGA-B after 100 runs are not reliable, their solution qualities are obviously inferior to those gotten by the other three approaches. On the other hand, the statistical values illustrate that CCPIA-A can obtain the best search performance by comparison with CCPIA-P and CCPIA-G. In addition, the values on AT illustrate that SSGA-A and SSGA-B need a lot of runtime to solve the above high-dimensional CCP problem, whereas other approaches spend only less time to find the desired solutions. Thereby, CCPIA can efficiently solve one such high-dimensional problem, in which the adaptive sampling method can speed up to seek the desired solution.

6 Conclusions and further work

Our theoretical work in this paper includes two aspects. One is to give a theoretical result which estimates the lower bound of sample size of a random vector appearing in a given CCP problem; the other is to propose a simple CCPIA for CCP problems with unknown noisy information, where some immune metaphors give us inspirations in constructing the immune modules. In CCPIA, the schemes of sample allocation and evaluation can not only effectively decide the importance of evolving B cells, but also reduce CCPIA′s computational cost. Additionally, a suppression radius function is designed to suppress noises dynamically, while the idea of fitness inheritance is utilized to accelerate the process of optimization. This way, our optimization

Table 6 Comparison of statistic values found for Example 5

	Opt.	β	*Mean*	*Std.Dev*	*CI*	*SR*(%)	*CR*	*RER*(%)	*AT*
SSGA-A			271.771	58.449 3	[270.2, 273.341]	1	0.289	25.3	94.1
SSGA-B			216.364	78.931 6	[214.243, 218.484]	15	23.02	40.5	93.4
CCPIA-A	363.925	0.87	312.016	24.110 6	[311.368, 312.664]	100	0	14.2	19.1
CCPIA-G			268.364	9.5416 2	[268.108, 268.621]	100	0	26.2	17.29
CCPIA-U			238.725	11.582 6	[238.413, 239.036]	99	10^{-4}	34.4	20.4
SSGA-A			305.858	46.58	[305.07, 306.646]	4	0.132	20.0	95.38
SSGA-B			252.123	70.501 9	[250.931, 253.316]	3	0.129	34.05	94.92
CCPIA-A	382.345	0.97	311.597	24.113 7	[311.189, 312.005]	100	0	18.5	19.43
CCPIA-G			267.552	10.415 4	[267.376, 267.728]	100	0	30.0	17.96
CCPIA-U			233.641	12.346	[233.433, 233.85]	100	0	38.9	21.7

mechanism does not need to know noisy information, while being capable of discovering the approximate optimum for a given CCP problem. Experimental results show that even if the noise is strong, CCPIA can also display many merits, e.g., low computational complexity and good effect. Comparative experiments have demonstrated that different mutation rules make CCPIA exhibit different characteristics; CCPIA-A and CCPIA-U perform well over other approaches, and CCPIA-G is secondary. Especially, CCPIA-A is a potential tool for complex high-dimensional CCP problems. We will aim further at studying its theoretical foundations and wide applications.

Appendix

Proof of Theorem 1. If $\boldsymbol{x}_\alpha^* \in X_M^\beta$, we obtain that $F_M(\boldsymbol{x}_\alpha^*) \geq \widehat{\theta}_M^\beta$. Hence, it follows from $\theta_\alpha^* < \widehat{\theta}_M^\beta - \eta$ that $F_M(\boldsymbol{x}_\alpha^*) - \mathrm{E}[f(\boldsymbol{x}_\alpha^*, \xi)] > \eta$, which, by means of the Hoeffding′s inequality, yields that

$$\begin{aligned} &Pr\{\theta_\alpha^* < \widehat{\theta}_M^\beta - \eta | \boldsymbol{x}_\alpha^* \in X_M^\alpha\} \leq \\ &\quad Pr\{F_M(\boldsymbol{x}_\alpha^*) - \mathrm{E}[f(\boldsymbol{x}_\alpha^*, \xi)] > \eta\} \leq \\ &\quad 2\mathrm{e}^{-\frac{2M\eta^2}{(b-a)^2}} \leq 2\mathrm{e}^{-2Mk^2}. \end{aligned} \tag{8}$$

On the other hand, let

$$X_M^{\beta_i} = \{\boldsymbol{x} \in X | \frac{1}{M} \sum_{l=1}^{M} \Pi(G_i(\boldsymbol{x}, \xi^{l,i}) > 0) \leq \beta_i\}. \tag{9}$$

Since $\boldsymbol{x}_\alpha^*$ satisfies the deterministic constraints as in P_α, $\boldsymbol{x}_\alpha^* \in X_M^\beta$ if and only if $\boldsymbol{x}_\alpha^* \in X_M^{\beta_i}$ with $1 \leq i \leq I$. Moreover, it is true that $\boldsymbol{x}_\alpha^* \in X_M^{\beta_i}$ if and only if $\sum_{l=1}^M \Pi(G_i(\boldsymbol{x}, \xi^{l,i}) > 0) \leq M\beta_i$. If we call the event $\{G_i(\boldsymbol{x}_\alpha^*, \xi^{l,i}) > 0\}$ a success for given l and i, the probability of a success in trial l is $p_i(\boldsymbol{x}_\alpha^*) = Pr\{G_i(\boldsymbol{x}^*, \xi^{l,i}) > 0\} \leq \alpha_i$. This occurs that

$$\begin{aligned} &Pr\{\boldsymbol{x}_\alpha^* \in X_M^\beta\} = \prod_{i=1}^{I} Pr\{\boldsymbol{x}_\alpha^* \in X_M^{\beta_i}\} = \\ &\prod_{i=1}^{I} \rho(\beta_i, p_i, M) \geq \prod_{i=1}^{I} \rho(\lfloor \beta_i M \rfloor, \alpha_i, M) \end{aligned} \tag{10}$$

where $\rho(\alpha_i, \beta_i, M)$ represents the probability of having at most $\lfloor \alpha_i M \rfloor$ success in M independent trials[29]. Further, owing to $\beta > \alpha$, the Chernoff inequality[30] implies that

$$\begin{aligned} &\rho(\lfloor \beta_i M \rfloor, \alpha_i, M) \geq 1 - \mathrm{e}^{-\frac{M(\beta_i - \alpha_i)^2}{2\alpha_i}} \geq \\ &\quad 1 - \mathrm{e}^{-2Mk^2}. \end{aligned} \tag{11}$$

Therefore, it follows from (10) and (11) that

$$\begin{aligned} &Pr\{\boldsymbol{x}_\alpha^* \notin X_M^\beta\} \leq 1 - \{1 - \mathrm{e}^{(-2Mk^2)}\}^I \leq \\ &\quad c \times \mathrm{e}^{-2Mk^2}, \quad \text{with } c > I. \end{aligned} \tag{12}$$

This way, (8) and (12) imply that

$$\begin{aligned} &Pr\{\theta_\alpha^* < \widehat{\theta}_M^\beta - \eta\} = Pr\{\theta_\alpha^* < \widehat{\theta}_M^\beta - \eta | \boldsymbol{x}_\alpha^* \in X_M^\alpha\} \times \\ &\quad Pr\{\boldsymbol{x}_\alpha^* \in X_M^\alpha\} + Pr\{\theta_\alpha^* < \widehat{\theta}_M^\beta - \\ &\quad \eta | \boldsymbol{x}_\alpha^* \notin X_M^\alpha\} Pr\{\boldsymbol{x}_\alpha^* \notin X_M^\alpha\} \leq \\ &\quad Pr\{\theta_\alpha^* < \widehat{\theta}_M^\beta - \eta | \boldsymbol{x}_\alpha^* \in X_M^\alpha\} + Pr\{\boldsymbol{x}_\alpha^* \notin X_M^\alpha\} \leq \\ &\quad (c+2)\mathrm{e}^{\{-2Mk^2\}}. \end{aligned} \tag{13}$$

□

Proof of Theorem 2. For a given current population X with size N at the moment n, the module of the evaluation evaluates a_{n1} times for the B cells in X given by

$$\begin{aligned} &a_{n1} = N_1(\min\{M, \frac{M_n}{N_1}\} + \min\{M, \frac{M_n}{N_1}\} I + J + K) + \\ &\quad (N - N_1)(\min\{M, \frac{M_n}{N}\} + \min\{M, \frac{M_n}{N}\} I + J + K) \leq \\ &\quad N(MI + J + K). \end{aligned} \tag{14}$$

On the other hand, the adaptive mutation evaluates at most a_{n2} times for the N clones mutated, given by

$$a_{n2} = N((M_c + 1)I + J + K). \tag{15}$$

Hence, within a run period, the sum of evaluations of the appearing individuals a_n can be obtained by unifying (4), (5) and (12),

$$\begin{aligned} a_n = a_{n1} + a_{n2} \leq N(M_\delta I + J + K) + \\ N((M_c + 1)I + J + K). \end{aligned} \tag{16}$$

Further, since the sizes of I, J and K usually are small, it follows from $M_c < M_\delta$ that $a_n = O(NM_\delta)$. □

References

[1] Y. Jin, J. Branke. Evolutionary optimization in uncertain environments – a survey. *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pp. 303–317, 2005.

[2] L. H. Lee, N. A. Pujowidianto, L. W. Li, C. Chen, C. M. Yap. Approximate simulation budget allocation for selecting the best design in the presence of stochastic constraints. *IEEE Transactions on Automatic Control*, vol. 57, no. 11, pp. 2940–2945, 2012.

[3] A. Nemirovski, A. Shapiro. Convex approximations of chance constrained programs. *SIAM Journal on Optimization*, vol. 17, no. 4, pp. 969–996, 2006.

[4] D. L. Olson, S. R. Swenseth. A linear approximation for chance-constrained programming. *Journal of the Operational Research Society*, vol. 38, no. 3, pp. 261–267, 1987.

[5] L. Zhang, L. Wang, D. Z. Zheng. Hypothesis-test based genetic algorithm for stochastic optimization problems. *Control Theory and Applications*, vol. 21, no. 6, pp. 885–889, 2004.

[6] A. K. Kahng, B. Liu, Q. K. Wang. Stochastic power/ground supply voltage prediction and optimization via analytical placement. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 8, pp. 904–912, 2007.

[7] A. Di Pietro, L. While, L. Barone. Applying evolutionary algorithms to problems with noisy, time-consuming fitness functions. In *Proceedings of Congress on Evolutionary Computation*, IEEE, Portland Oregon, USA, vol. 2, pp. 1254–1261, 2004.

[8] N. V. Sahinidis. Optimization under uncertainty: State-of-the-art and opportunities. *Computers & Chemical Engineering*, vol. 28, vol. 6–7, pp. 971–983, 2004.

[9] H. Agarwal. Reliability Based Design Optimization: Formulations and Methodologies, Ph. D. dissertation, University of Notre Dame, USA, 2004.

[10] K. Deb, S. Gupta, D. Daum. Reliability-based optimization using evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1054–1074, 2009.

[11] J. Luedtke. A branch-and-cut decomposition algorithm for solving chance-constrained mathematical programs with finite support. *Mathematical Programming*, vol. 146, no. 1–2, pp. 219–244, 2013.

[12] F. F. Dong, Y. Liu, L. Qian, H. Sheng, Y. H. Yang, H. C. Guo, L. Zhao. Interactive decision procedure for watershed nutrient load reduction: An integrated chance-constrained programming model with risk-cost tradeoff. *Environmental Modelling & Software*, vol. 61, pp. 166–173, 2014.

[13] C. B. Wu, G. H. Huang, W. Li, Y. L. Xie, Y. Xu. Multi-stage stochastic inexact chance-constraint programming for an integrated biomass-municipal solid waste power supply management under uncertainty. *Renewable and Sustainable Energy Reviews*, vol. 41, pp. 1244–1254, 2015.

[14] C. W. Cao, X. S. Gu, Z. Xin. Chance constrained programming models for refinery short-term crude oil scheduling problem. *Applied Mathematical Modelling*, vol. 33, no. 3, pp. 1696–1707, 2009.

[15] H. Zhang, P. Li. Chance constrained programming for optimal power flow under uncertainty. *IEEE Transactions on Power Systems*, vol. 26, no. 4, pp. 2417–2424, 2011.

[16] K. Ağpak, H. Gókçen. A chance-constrained approach to stochastic line balancing problem. *European Journal of Operational Research*, vol. 180, no. 3, pp. 1098–1115, 2007.

[17] M. T. Rantanen, M. Juhola. A configuration deactivation algorithm for boosting probabilistic roadmap planning of robots. *International Journal of Automation and Computing*, vol. 9, no. 2, pp. 155–164, 2012.

[18] C. A. Poojari, B. Varghese. Genetic algorithm based technique for solving chance constrained problems. *European Journal of Operational Research*, vol. 185, no. 3, pp. 1128–1154, 2008.

[19] Y. Tenne. An optimization algorithm employing multiple metamodels and optimizers. *International Journal of Automation and Computing*, vol. 10, no. 3, pp. 227–241, 2013.

[20] L. N. de Castro, J. Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach*, Berlin, Germany: Springer-Verlag, 2002.

[21] E. Hart, J. Timmis. Application areas of AIS: The past, present and the future. *Applied Soft Computing*, vol. 8, no. 1, pp. 191–201, 2008.

[22] V. Cutello, G. Nicosia, M. Pavone. Real coded clonal selection algorithm for unconstrained global optimization using a hybrid inversely proportional hypermutation operator. In *Proceedings of the 21st Annual ACM Symposium on Applied Computing, SAC 2006*, ACM, Dijon, France, vol. 2, pp. 950–954, 2006.

[23] V. Cutello, G. Nicosia, M. Pavone, J. Timmis. An immune algorithm for protein structure prediction on lattice models. *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 1, pp. 101–117, 2007.

[24] D. Dasgupta, S. H. Yu, F. Nino. Recent advances in artificial immune systems: Models and applications. *Applied Soft Computing*, vol. 11, no. 2, pp. 1574–1587, 2011.

[25] Q. Y. Zhao, R. Yang, F. Duan. An immune clonal hybrid algorithm for solving stochastic chance-constrained programming. *Journal of Computational Information Systems*, vol. 8, pp. 8295–8302, 2012.

[26] Z. H. Zhang, L. Wang, M. Liao. Adaptive sampling immune algorithm solving joint chance-constrained programming. *Journal of Control Theory and Applications*, vol. 11, no. 2, pp. 237–246, 2013.

[27] F. O. de França, F. J. Von Zuben, L. N. de Castro. An artificial immune network for multimodal function optimization on dynamic environments. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, ACM, New York, USA, pp. 289–296, 2005.

[28] Z. H. Zhang, T. Xin. Immune algorithm with adaptive sampling in noisy environments and its application to stochastic optimization problems. *IEEE Computational Intelligence Magazine*, vol. 2, no. 4, pp. 29–40, 2007.

[29] J. Luedtke, S. Ahmed. A sample approximation approach for optimization with probabilistic constraints. *SIAM Journal on Optimization*, vol. 19, no. 2, pp. 674–699, 2008.

[30] B. K. Pagnoncelli, S. Ahmed, A. Shapiro. Sample average approximation method for chance constrained programming: Theory and applications. *Journal of Optimization Theory and Applications*, vol. 142, no. 2, pp. 399–416, 2009.

[31] F. M. Burnet. *The Clonal Selection Theory of Acquired Immunity*, Cambridge, UK: Cambridge University Press, 1959.

[32] K. Deb, M. Goyal. A combined genetic adaptive search (GeneAS) for engineering design. *Computer Science and Informatics*, vol. 26, pp. 30–45, 1996.

[33] B. Varghese, C. A. Poojari. Genetic algorithm based technique for solving chance constrained problems arising in risk management. Technical Report, Carisma, 2004.

Zhu-Hong Zhang received his M. Sc. degree from Department of Mathematics, Guizhou University, China in 1998 and Ph. D. degree from College of Automation, Chongqing University, China in 2004. He is a professor at Guizhou University, China. Prof. Zhang has been an associate editor affiliated with *Journal of Applied Soft Computing* since 2010. He has published about 80 refereed journal and conference papers and also one book on modern intelligent algorithms (the 2nd author).

His research interests include evolutionary computation, immune optimization, uncertain programming, control theory, and visual neural networks.

E-mail: sci.zhzhang@gzu.edu.cn (Corresponding author)

ORCID iD: 0000-0001-7619-1040

Kai Yang received his B. Sc. and M. Sc. degrees from Departments of Mechanical Engineering and Computer Science, Guizhou University, China in 1998 and 2008, respectively. He is currently a Ph. D. degree candidate at College of Computer Science, Guizhou University, China.

His research interests include immune optimization and stochastic programming.

E-mail: csc.kyang@gzu.edu.cn

Da-Min Zhang received his M. Sc. and Ph. D. degrees from Guizhou University, China in 2005 and 2010, respectively. He, as a professor at Guizhou University, has published about 30 refereed journal and conference papers.

His research interest include computer system integration, software development and complex networks.

E-mail: ie.dmzhang@gzu.edu.cn