Evolutionary Algorithms and Constraint Satisfaction: Definitions, Survey, Methodology, and Research Directions

1. Introduction

- *Why study constrained problem?*

  · Many problems are constrained.

  · Many untractable problems are constrained.

- *Why use EAs?*

  · good ratio of (implementation) effort to performance.

  · EAs are good solvers for tough problems.

- *no standard EA takes constraints into account.*

  · the regular search operators are 'blind' to constraints.

  · EAs perform unconstrained search.

2. What is a constrained problem?

-depends on what we take as the search space.

Def. 1. If, for each solution s in a search space S, $s' = \boldsymbol{mut}(s) \in S$, we call $\boldsymbol{mut}(*)$ *mut free.*

Def. 2. We call a recombination operator *free* with respect to the search space S if its application does not lead out of S.

Def. 3. We call a Cartesian product of sets $S = D_1 \times \ldots \times D_n$ a *free search space*. The domains can be continuous or discrete.

· Testing membership relation of a free search space can be done independently on each coordinate.

· Usual mutation and recombination operators are free w.r.t. a free search space.

Def. 4. A *free optimization problem* (FOP) is a pair $<S, f>$, where S is a free search space and f is a (real-valued) optimization function on S. A *solution* of a FOP is an $s \in S$ with an optimal f-value.

Def. 5. A *constraint satisfaction problem* (CSP) is a pair $<S, \Phi>$, where S is a free search space and Φ is a formula (Boolean function on S). A *solution* of CSP is an $s \in S$ with $\Phi(s) = true$.

Def. 6. A *constrained optimization problem* (COP) is a pair $<S, f, \Phi>$, where S is a free search space, f is a (real-valued) optimization function on S and Φ is a formula. A *solution* of COP is an $s \in S$ with $\Phi(s) = true$ and an optimal f-value.

- For CSP and COP, we call Φ the *feasibility condition*, and the set $\{s \in S \mid \Phi(s) = true\}$ the *feasible search space*.

- Solving a CSP means finding a feasible element, and solving a COP means finding a feasible and optimal element.

3. Constraint handling in an EA

- three ways to handle constraints

  · handle all constraints directly

  · handle all constraints indirectly

  · handle some of the constraints directly and other ones indirectly

  · which constraints should be handled directly and which should be handled indirectly?

-direct handling

  · leave the constraints as they are and *modify EA to enforce them*.

  · how to enforce the constraints?

-indirect handling

  · *transform constraints into optimization objectives*, thus relaxing ϕ and modifying f (in case of a COP) or introducing f (in case of a CSP).

  · how to transform the constraints?

3.1 Direct constraint handling

- have to create and maintain feasible chromosomes in the population.

-eliminating infeasible candidates

  · very inefficient (the probability of hitting a feasible candidates by chance is practically zero).

-repairing infeasible candidates

  · problem dependent (requires repair procedure).

-preserving feasibility by special operators

  · design and apply problem specific operators that do reserve the feasibility of parent chromosomes

  · quasi-free (if parents are feasible, the offspring will be feasible)

-decoding, i.e., transforming the search space

  · elements of the new search space serve as inputs for a decoding procedure.

  · free search can be performed by EA in the new search space.

  · can simplify the problem and allow an efficient EA.

- heavily problem dependent, but work well

3.2 Indirect constraint handling

-penalty function

  · the optimization objectives replacing the constraints

  · estimate of the badness of a given candidate solution.

  · penalty for violated constraints:

$$f_1(s) = \sum_{i=1}^{m} w_i \, c(s, c_i) \text{ with } c(s, c_i) = \begin{cases} 1 \text{ if } s \text{ violates } c_i \\ 0 \quad \text{otherwise} \end{cases}$$

  · penalty for wrongly instantiated variables:

$$f_2(s) = \sum_{i=1}^{n} w_i c(s, C^i) \text{ with } c(s, C^i) = \begin{cases} 1 \text{ if } s \text{ violates at least one } c \in C^i \\ 0 \text{ otherwise} \end{cases}$$

· $d(s)$ = *true* if and only if $f_i(s) = 0$.

-advantages

· there exist problem independent realization of the principle.

· reduces the problem to simple optimization

· allows a way to express user preferences distinguishing among constraints

-disadvantages

· loss of information by packing all knowledge on constraint violation into a single number

· does not work well for sparse problems, where the proportion of feasible solutions is very low

· problem dependent issues: defining optimization objectives, merging the original objective function and penalties (in case of a COP)