Experimental Results
for Population Initialization

9.1 Overview

In Chap. 5 we suggested that the sentinel placement algorithm that we have
developed for our dynamic environment EA might also be useful for popu­
lation initialization for EAs in static environments. In this chapter, and also
in [43], we examine that suggestion by testing a standard EA in a variety
of static landscapes using random population initialization and population
initialization using the sentinel placement algorithm (referred to as “placed
initialization”).

9.2 Background

EAs are stochastic search techniques. As such, normal execution of an EA
requires many runs to provide reasonable assurance that any negative effects
of merely “bad luck” in the stochastic processes have been overcome. One
of the first stochastic influences on the behavior of an EA is the population
initialization. This has been recognized as a potentially serious problem to the
performance of EAs in [32], and, to a lesser extent, in [41] and [12], but little
progress has been made in improving the situation. In the few cases where this
situation is addressed at all, populations, in very low-dimensionality problems,
have been initialized using mathematical techniques like the Latin hypercube
[12]. The Latin hypercube technique guarantees uniform placement along each
axis, but as was shown in Chap. 5, uniform placement along individual axis
projections does not ensure any level of uniformity throughout the search
space. In other words, the Latin hypercube technique would provide good
values for our intermediate dispersion calculation $S_1$, but says nothing about
the value of the calculation $S_2$, so it does not ensure uniform search-space
coverage, as would be measured by $\Delta$.

Since EAs are generally executed many times to overcome any negative ef­
teffects of, among other things, the possibility of a bad population initialization,
using a better population initialization algorithm would not be expected to improve the average performance of an EA (if executed many times). Instead, it would be expected to reduce the variance without loss of average performance, thereby providing researchers with the opportunity to reliably examine their experimental results while requiring fewer EA runs for an appropriate statistical sample. One would also expect the reduction in variance to be most noticeable in the early generations of the EA run. As was described in Chap. 5, population initialization using the sentinel placement algorithm instead of randomization is not expected to be useful for problems of dimensionality greater than 12 due to the large spatial distances involved. It needs to be reiterated that we are focusing here on average performance. It is conceivable that for some specific problems, more uniform population initialization might actually reduce the peak algorithm performance.

9.3 Experiment

To examine the usefulness of the sentinel placement algorithm for population initialization, a simple EA with gray code binary representation, fitness-proportional selection, uniform crossover, and a mutation rate of 0.001 was used. This EA was run against seven different problems of varying complexity. These problems are:

- De Jong test function #2, also called Rosenbrock’s function, in two dimensions [37].
- De Jong test function #3, a step function, in five dimensions [37].
- Michalewicz’s function, in ten dimensions [37].
- The 5-cone static landscape used for the dynamic problems described in Chap. 6, in two dimensions.
- The 14-cone static landscape used for the dynamic problems described in Chap. 6, in two dimensions.

A population of 25 was used for the De Jong and Michalewicz test functions and, as in the dynamic landscape problems, a population of 50 was used for the conical landscapes. The De Jong test functions #2 and #3 and the Michalewicz test function are usually implemented as minimization functions, but were implemented as maximization functions for these experiments, applying the following equations respectively:

\[ f(\bar{x}) = 3907 - 100(x_1^2 - x_2)^2 + (1 - x_1)^2, \text{ for } x_{1,2} \in [-2.048, 2.048]. \]  
(9.1)

\[ f(\bar{x}) = 25.0 - \sum_{i=1}^{N} |x_i|, \text{ for } x_i \in [-2.048, 2.048]. \]  
(9.2)