

# Measuring the Searched Space to Guide Efficiency: The Principle and Evidence on Constraint Satisfaction

Jano van Hemert  
Leiden University  
jvhemert@liacs.nl

Thomas Bäck  
NuTech Solutions GmbH & Leiden University  
baeck@liacs.nl

## Searched Space

### DEFINITION 1

State space: The total *state space* an evolutionary algorithm is able to search in is defined as  $M = D_1 \times \dots \times D_l$ . We assume all  $D_i$  are equal, thus we know that  $M = D^l$  and  $|M| = |D|^l = m^l$ .

### DEFINITION 2

Searched space of an algorithm: The *searched space*  $S \subset M$  is the set of points taken from the state space that is visited by a particular evolutionary algorithm during a run.

### DEFINITION 3

Resampling ratio: First we define a *revisit* as a point in the state space that we have seen before, i.e., it is already present in the searched space. The *resampling ratio* is defined as the total number of revisits in a run divided by the total number of evaluations in the same run:  $\text{resampling ratio} = \text{revisits}/\text{evaluations}$ .

### HYPOTHESIS 1

A resampling ratio too high will have a negative influence on the performance of an algorithm.

### HYPOTHESIS 2

The mutation operator with mutation rate  $k/l$  has a large impact on the searched space, especially for small values of  $k$ .

## The Chance of Not Changing

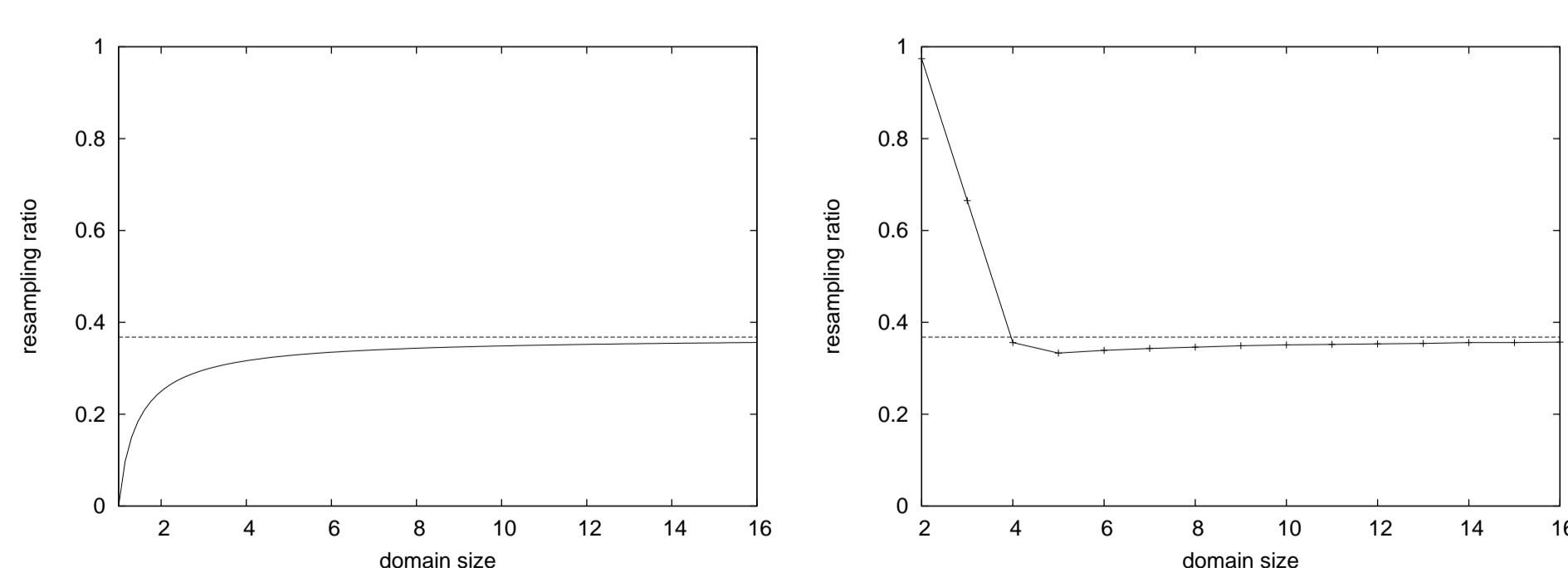
A mutation operator with  $p_m = k/l$  on a chromosome  $c$  of length  $l$ :

for  $i = 1$  to  $l$  do  
if (uniform-random( $l$ )  $\leq k$ ) then  
 $c_i = \text{uniform-random}(D - c_i)$

$\text{resampling ratio} = P(\text{mutate}(c) = c)$

$$P(\text{mutate}(c) = c) = (1 - p_m)^l = \left(\frac{l - k}{l}\right)^l$$

$$\lim_{l \rightarrow +\infty} \left(\frac{l - k}{l}\right)^l = e^{-k}$$



The chance of not changing a chromosome (left). Simulation of the mutation operator with  $p_{\text{mutation}} = 1/l$ , thus  $k = 1$  (right).

## Practical Evidence with Binary Constraint Satisfaction Problems

### DEFINITION 4

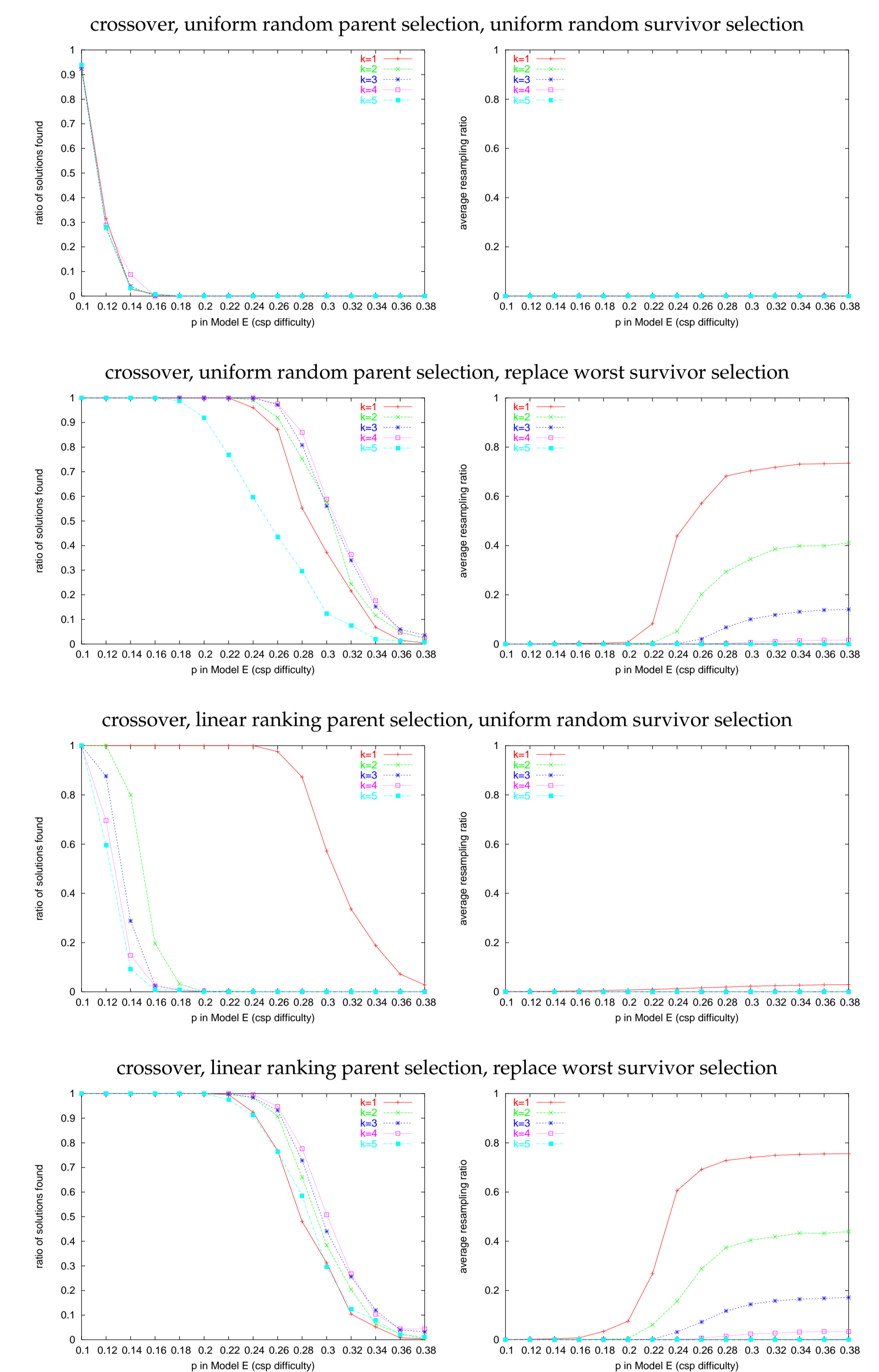
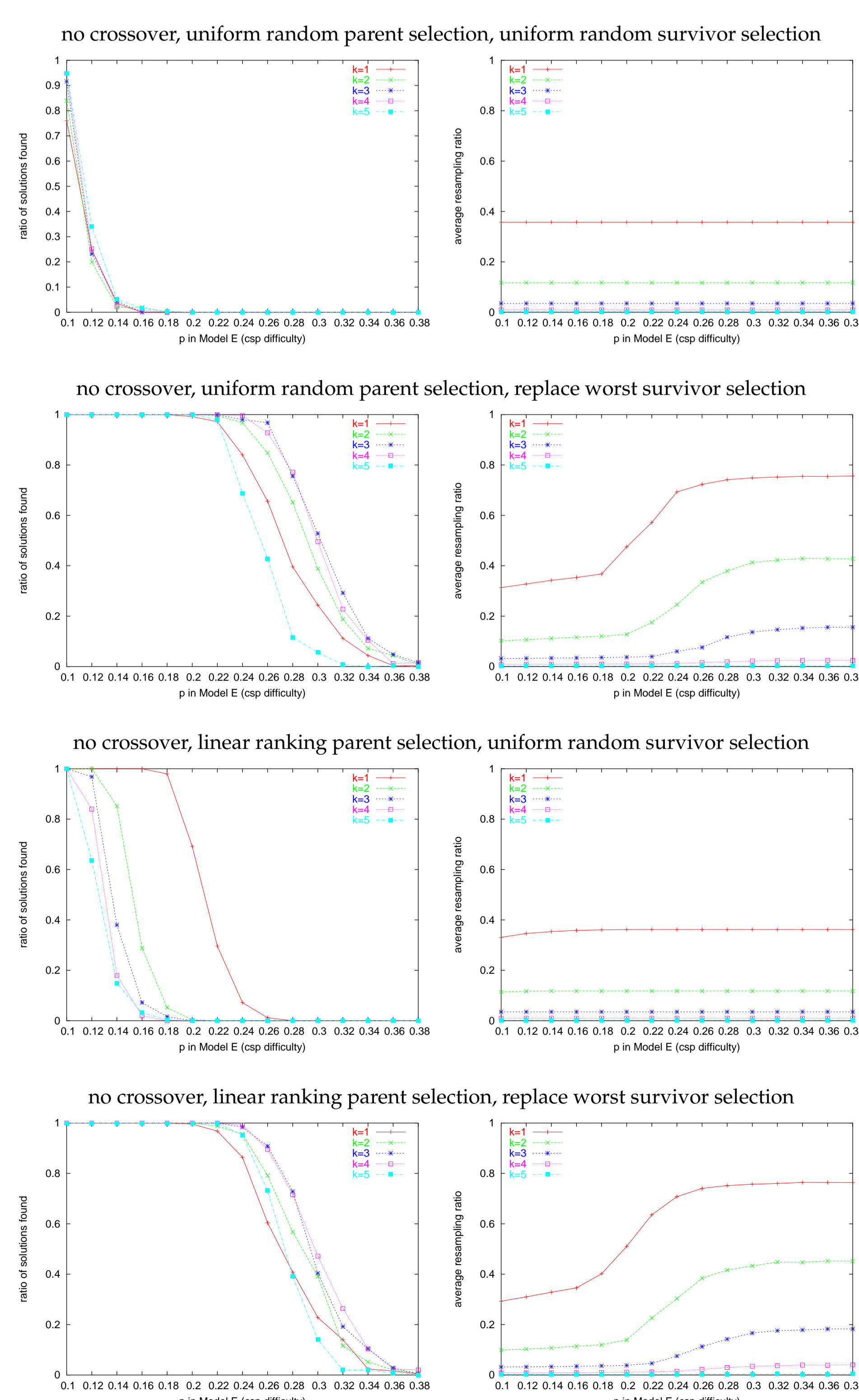
A Binary Constraint Satisfaction Problem (CSP) is a tuple  $\langle X, D, C \rangle$  where

- $X$  is a set of variables,
- $D$  is a set of finite domains  $\{D_{x_1}, D_{x_2}, \dots, D_{x_{|X|}}\}$ ,
- and  $C$  is a set of constraints that restrict certain simultaneous object assignments, where each of  $c \in C$  is over exactly two variables.

### Experimental setup

- We randomly generate binary CSPs using Model E  $\langle 15, 15, p \rangle$ , where we vary the parameter  $p$  from 0.10 to 0.38. ( $l = |X| = 15$  and  $\forall x_i \in X : |D_{x_i}| = 15$ )
- We use 25 unique problem instances for every setting of  $p$  and run 10 tests on each instance, with a maximum of 100,000 evaluations.
- The success rate and the resampling ratio are measured.
- Crossover, parent selection and survivor selection are switched on and off.
- We test five mutation rates ( $k = 1, \dots, 5$ ) for each combination.

### Results



## Conclusions

- In our experiments, a small resampling ratio (0.5–3%) corresponds with high success rates.
- A low resampling ratio on itself is not a good prediction for good performance.
- Whenever an evolutionary algorithm's resampling ratio increases this is a good indication that its efficiency and its effectiveness is decreasing.
- When using the replace worst survivor selection a mutation rate of  $4/l$  is the best.

## Future Work

- Can we accurately predict the influence of  $k$  and different selection processes?
- Can we generalise what a "healthy" resampling ratio is?
- Are we able to sustain a "healthy" resampling ratio during a run?