Problem Space formulation

1 Introduction

The following is an explanation of the formal terms used in defining and identifiying the cuts used to generate the plots for re-compare library.

Definition (Solution)**.** *Given a general problem F which is takes parameters $p_i \in P_i \mid 1 \leq i \leq n$ and whose possible solution can be represented in a tuple of independant values $s_i \in S_i \mid 1 \leq j \leq m$, the problem can be seen as a function*

$$f : P \to 2^S$$
$$P = \times_{i=1}^{n} P_i$$
$$S = \times_{j=1}^{m} S_i$$

we would say that $s \in S$ is a solution of $p \in P$ if $s \in f(p)$.

Definition (Problem space)**.** *Using the notations above, we mark P as the parameter space, S as the solution space and $Pr =: P \times S$ as the solution space. We define $\{(p, s)) \mid p \in P, s \inf(p)\}$ as the proper (feasible) problem space*

When analyzing high dimensional information, we often time try look for a dependancy between some P_i and some S_j when leaving all other $P_k \mid k \neq i$ constants. This defines a sub space on the problem space.

Definition (Cuts)**.** *A cut of the problem space of a problem f is a sub space the proper solution space*

$$\pi_{k_1=c_1,\ldots,k_r=c_r,k'_1=d_1,\ldots,k'_{r'}=d'_r}(Pr) = \{(p, f(p)) \mid p_{k_i} = c_i, s_{k'_j} = d_j\}$$

Basically, we take only the proper tuples that agreed with the designated constant equality constraints we put on some of the dimensions of the problem space.

We define a k-cut as a cut that is not constrained on k dimensions. We define a r_1, r_2-cut as a subspace that in not constrained only on dimensions r_1 and r_2. This definition generalises to any amount of dimensions.

Analysing and Visualising

When trying to analyse a Problem space by experimental computations and present results in a manner thats is informative for a human, we will present 3 dimensional cuts of the problems space as 2 dimensional plots as **discrete layer plots**.

Definition (DLPs (Discrete layer plots))**.** *A DLP is a tuple of the form (Q_1, Q_2, F, C_1, C_2) Where*

- *Q_1, Q_2, F are dimensions of the problem space.*
- *C_1, C_2 are discrete subsets of Q_1, Q_2 respectivly.*

We visualize the DLP on a 2-dimensional plot by taking

- Q_1 to be the independant variable on the x axis
- F to be the dependant variable on the y axis
- Q_2 to be another independant variable that varies across layers of the plot

The discretness restriction on Q_2 stems from a visualization constraint (since we cant have a continuum of layers in a plot) and both independant parameters are constrained because we are sampling Algorithms on finite sets of inputs.

Localization to Re-comp

In Re-comp we compare algorythms of the type $match(text, patern)$ and are usually interested in the time performance of the matching operations. Thus, for a set of Algs A and a space of texts and regex patterns T, R we would define the Problem space as

$$P = A \times T \times R$$
$$S = \mathbb{R} \cong time$$

Since we are interested in seeing how different algorythms fair in time across varying texts and patterns, we are interested in DLPs of the form $(T_i, A, time, C, A)$ or $(R_i, A, time, C, A)$. This basicly means that we show all the results in terms of $time$ keeping all parameters of the problems constant except the algorithm chosen and a single parameter of the regexes or texts.

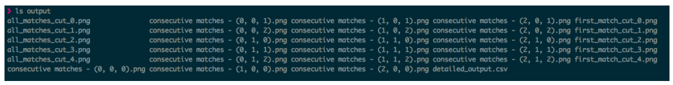

Fig. 1: An example of DLP plot from recompare

What makes re-compare stands out is that it computes all such DLPs given any parametrisation to the regex space and text space.