Distance Function

Distance function $d : s \times s \rightarrow \mathcal{R}$ calculates the distance or differences between two solutions s_1, s_2 into real numbers $\mathcal{R}$. Given solutions s_1, $s_2 \in S(\pi)$, a distance function must follow the following axioms of the metric space:

1. $d(s_1, s_2) \geq 0$
2. $d(s_1, s_2) = 0$ when $s_1 = s_2$, otherwise $d(s_1, s_2) > 0$
3. $d(s_1, s_2) = d(s_2, s_1)$ (symmetric)
4. $d(s_1, s_3) \leq d(s_1, s_2) + d(s_2, s_3)$ (triangle inequality)

The triangle inequality is important as it corresponds to spatial intuition when visualized, thus allowing us to transform the search space into abstract 2-D fitness landscape in our FLST visualization (see Chapter **??**).

There are several distance functions which meet these requirements (see [3, 5, 6, 1, 4, 2, 7]). In this section, we describe three of them, which are used quite often in our experiments:

Hamming/Exact distance

This distance function is used for measuring the distance of two permutation/bit string/domain-based based solutions where the position/order of the items of the solutions matter because the solution array represents the assignment, e.g. QAP, LABS, MKP. We want to know the number of mismatch between each index position in the two solutions. This distance function can be computed in one linear pass — $O(n)$. The maximum value for this distance function is n.

$d(s_1, s_2) = \sum_{k=0}^{n-1} \texttt{mismatch\_k}$
$\texttt{mismatch\_k} =$
$\texttt{1, if } s_{1_k} \neq s_{2_k}$
$\texttt{0, otherwise}$
$d_{max} = n$

Bond/Permutation/Edge distance

This distance function is used for measuring the distance of two permutation based solutions where the order of items does not matter but the edges/links between the items matter, e.g. TSP, variants of VRP. We want to count the number of different edges between the two solutions. This distance function can be computed in one linear pass — $O(n)$ with appropriate data structure. The maximum value for this distance function is n.

$d(s_1, s_2) = n - \sum_{i=0}^{n-1} \texttt{common\_edges\_k}$
$\texttt{common\_edges\_k} =$
$\texttt{1, if } \exists l \in [0 \dots n-1] \rightarrow \texttt{e}(s_{2_l}, s_{2_{(l+1)\%n}}) = \texttt{e}(s_{1_k}, s_{1_{(k+1)\%n}}) \texttt{ or } \texttt{e}(s_{1_{(k+n-1)\%n}}, s_{1_k})$
$\texttt{0, otherwise}$
$d_{max} = n$

Deviation distance

This distance function is used to measure the deviation of the position of items, e.g. JSSP. This distance function can be computed in linear pass — $O(n)$ provided we preprocess the indices first. In the original form, the maximum value for this distance function is $n * (n-1)$. However, to make it similar with the rest, we divide it with $n-1$ so that the maximum value remains n.

$d(s_1, s_2) = (\sum_{k=0}^{n-1} \texttt{deviation\_k})/(n-1)$
$\texttt{deviation\_k} = |k_1 - k_2|$ `where` $s_{1_{k_1}} = s_{2_{k_2}}$
$d_{max} = n$

Distance information derived from distance function is used by the SLS algorithm itself (e.g. measuring population diversity in GA [7, 5, 6]) or by analysis tool that requires distance information such as our FLST visualization (see Chapter **??** and Chapter **??**).

The choice of a particular distance function depends on the nature of the COP being attacked and the elements that constitute a meaningful notion of distance within the fitness landscape, otherwise the results will be inaccurate, e.g. bond distance is appropriate for TSP but not for QAP as edges in QAP solutions does not mean anything. Similarly, if we use Hamming distance to measure distance between TSP solutions, the resulting distances will be too far, misleading our observation (tour 1-2-3-4 and tour 2-3-4-1 are essentially the same tour but Hamming distance will report distance = 4 in this case).

As distance function is frequently used in FLST visualization, its computation must be done as fast as possible! Currently, all the distance functions shown above can be computed in $O(n)$ (linear time) by using appropriate data structures.

The resulting values reported by a distance function can be 'normalized' into $[0 \ldots 1]$ by dividing the results with d_{max}. In Table 1, Table 2, and Table 3, we give examples of the usage of distance functions shown in this Appendix .

		$d(s_1, s_2)$	d_{max}	$d(s_1, s_2)^{norm}$
s_1	`[1,0,1,1,0]`			
s_2	`[1,1,0,1,0]`			
$d(s_1, s_2)$	`0+1+1+0+0`	2	5	0.4

Table 1: Example of Hamming distance for solutions represented as bit string.

		$d(s_1, s_2)$	d_{max}	$d(s_1, s_2)^{norm}$
s_1	`[4,0,3,2,2]`			
s_2	`[4,1,3,1,0]`			
$d(s_1, s_2)$	`0+1+0+1+1`	3	5	0.6

Table 2: Example of Hamming distance for solutions represented as domain.

		$d(s_1, s_2)$	d_{max}	$d(s_1, s_2)^{norm}$
s_1	`[1,2,0,4,3]`			
s_2	`[4,1,2,0,3]`			
$d_1(s_1, s_2)$	`1+1+1+1+0`	4	5	0.8
$d_2(s_1, s_2)$	`5-|{(1,2),(2,0),(4,3)}|`	2	5	0.4
$d_3(s_1, s_2)$	`(1+1+1+3+0)`	1.5	5	0.3

Table 3: Example of Hamming/Exact (d_1), Bond/Permutation/Edge (d_2), and Deviation distance (d_3) for solutions represented as permutation.

Bibliography

[1] Cyril Fonlupt, Denis Robilliard, Philippe Preux, and El-Ghazali Talbi. Fitness Landscapes and Performance of Meta-heuristics. In *Meta-Heuristics - Advances and Trends in Local Search Paradigms for Optimization*, pages 255–266. Kluwer Academic Publishers, 1999.

[2] Holger H. Hoos and Thomas Stützle. *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann, 2005.

[3] Terry Jones. *Evolutionary Algorithms, Fitness Landscapes and Search*. PhD thesis, The University of New Mexico, Albuquerque, New Mexico, 1995.

[4] Peter Merz. *Memetic Algorithms for Combinatorial Optimization: Fitness Landscapes & Effective Search Strategies*. PhD thesis, University of Siegen, Germany, 2000.

[5] Simon Ronald. Distance functions for order-based encodings. In *IEEE International Conference on Evolutionary Computation*, pages 43–48, 1997.

[6] Simon Ronald. More distance functions for order-based encodings. In *IEEE International Conference on Evolutionary Computation*, pages 558–563, 1998.

[7] Marc Sevaux and Kenneth Sörensen. Permutation Distance for Memetic Algorithm. In *6th Metaheuristics International Conference*, 2005.

Index