A CONCEPTUAL MODEL FOR OPTIMIZATION PROBLEMS

JEAN P. MARTINS
jeanp@dei.uc.pt

Abstract. Defines general concepts related to combinatorial optimization problems and illustrates how they can be used to implement metaheuristics in terms of elementar neighborhood-based operations.

1. Definitions

Definition 1.1 (Component). *In a combinatorial optimization problem, a component* $c \in C$ *is the elementary unity of a solution.*

Definition 1.2 (Search space). *The search space* S *of a combinatorial optimization problem is a subset of the power set of components* $S_{\leq n}(2^C)$ *(only* S *from now on), in which elements* $x \in S$ *can be composed of a maximum of* n *components. In practical terms,* S *is implicitly defined by* C *and* n*.*

Example 1.1 (Knapsack problem component). *In a knapsack problem, a component is an item. There are* n *available items (components)* $C = \{1, 2, \ldots, n\}$*. Therefore, elements* $x \in S$ *can be composed of a maximum of* n *items, i.e.* $|C| = n$*.*

Example 1.2 (Minimum spanning tree component). *In a minimum spanning tree problem, a component is an edge. Considering a undirected complete graph* $G = (V, E)$*, with* $|V| = n$*, there are* $|C| = (n^2 - n)/2$ *available edges (components).*

Definition 1.3 (Neighborhood). *A neighborhood function* $N_k(x) = \{y \in S : d(x, y) = \delta(k)\}$ *defines the neighbors of* x *in terms of components, with* $k \geq 0$ *denoting the number of components in which* x *and* y *differ. The size of* $N(x)$ *is usually* $|C| \in O(n^k)$*.*

Example 1.3 (Add/remove neighborhood). *Solutions* $x, y \in S$ *are neighbors if they differ in only one component, i.e.* $k = 1$*. A solution* y *can be generated from* x *by adding or removing a component.*

Example 1.4 (Swap neighborhood). *Solutions* $x, y \in S$ *are neighbors if they differ in two components, i.e.* $k = 2$*. A solution* y *can be generated from* x *by removing a component and adding another component.*

Example 1.5 (2-opt neighborhood). *Solutions* $x, y \in S$ *are neighbors if they differ in two components, i.e.* $k = 2$*. A solution* y *can be generated from* x *by removing two components (edges)* (a, b) *and* (c, d) *and adding the components (edges)* (a, d) *and* (c, b)*.*

Observe that the neighborhoods add/remove and swap are independent of the component structure, whereas 2-opt assume the components are edges in a graph.

Date: July 7, 2015.

2. Implementation details

Definition 2.1 (Search space)**.** *The search space structure S defines the list of components C it supports.*

Definition 2.2 (Solution)**.** *Solutions x, as elements of the search space S, have a copy of the component list. The components composing x are stored in $C_u = \{c \in C : c \in x\}$, the remaining (available) are stored in $C_a = \{c \in C : c \notin x\}$. In summary, $C_u \cup C_a = C$.*

Definition 2.3 (Neighborhood)**.** *Neighboring solutions are generated by the function N, which has access to solutions x and their component lists.*