

Here's a Python implementation of the clustering algorithm to split the set of combinations into clusters of effective and non-effective combinations:

```
def create_effective_combinations(Cn, yc, xc):  
    """  
  
    Splits the set of combinations into effective and non-effective combinations.  
  
    Parameters:  
  
    Cn (list): List of n-combinations of drugs.  
  
    yc (dict): Dictionary with combinations as keys and their effectiveness as values.  
  
    xc (dict): Dictionary with combinations as keys and their side effect scores as values.  
  
    Returns:  
  
    dict: Dictionary with effective combinations for each n.  
    dict: Dictionary with non-effective combinations for each n.  
    """  
  
    effective_combinations = {}  
    non_effective_combinations = {}  
  
    for n in range(1, 8):  
        En = set()  
        NonEn = set()  
        combinations = [c for c in Cn if len(c) == n]  
  
        if not combinations:  
            continue  
  
        # Sort combinations by effectiveness  
        sorted_combinations = sorted(combinations, key=lambda c: yc[c], reverse=True)
```

```

# Initial selection based on the maximum yc
max_combination = sorted_combinations.pop(0)

En.add(max_combination)

ymax = yc[max_combination]
xmin = xc[max_combination]

while sorted_combinations:
    c = sorted_combinations.pop(0)
    if yc[c] > ymax:
        En.add(c)
        ymax = yc[c]
    elif xc[c] < xmin:
        En.add(c)
        xmin = xc[c]
    else:
        NonEn.add(c)

effective_combinations[n] = En
non_effective_combinations[n] = NonEn

return effective_combinations, non_effective_combinations

```

# Example usage

```

Cn = [('drug1',), ('drug2',), ('drug1', 'drug2'), ('drug1', 'drug3'), ('drug2', 'drug3'), ('drug1', 'drug2',
'drug3')]

yc = {('drug1',): 0.5, ('drug2',): 0.4, ('drug1', 'drug2'): 0.8, ('drug1', 'drug3'): 0.6, ('drug2', 'drug3'): 0.7,
('drug1', 'drug2', 'drug3'): 0.9}

xc = {('drug1',): 0.2, ('drug2',): 0.3, ('drug1', 'drug2'): 0.3, ('drug1', 'drug3'): 0.4, ('drug2', 'drug3'): 0.2,
('drug1', 'drug2', 'drug3'): 0.5}

```

```
effective_combinations, non_effective_combinations = create_effective_combinations(Cn, yc, xc)
print("Effective Combinations:", effective_combinations)
print("Non-Effective Combinations:", non_effective_combinations)
```

**Input Parameters:**

- Cn: List of drug combinations.
- yc: Dictionary where keys are combinations and values are effectiveness scores.
- xc: Dictionary where keys are combinations and values are side effect scores.

**Algorithm Steps:**

- Iterate over combination sizes from 1 to 7.
- For each size, initialize empty sets En and NonEn to store effective and non-effective combinations respectively.
- Sort the combinations based on their effectiveness (yc values) in descending order.
- Select the combination with the highest effectiveness as the initial effective combination.
- Update ymax to the effectiveness of this combination and xmin to its side effect score.
- Continue adding combinations to En or NonEn based on the criteria until all combinations are processed.