Task1

Method 1

Let's consider a specific approach using a combination of random sampling and incremental selection:

1. **Random Sampling**:
    - Randomly sample a small subset (let's say 1-5% of the total items) from the candidate set of 500 million items.
    - Evaluate the hypervolume contribution of each sampled item independently.
2. **Initial Subset Selection**:
    - Choose the item with the highest hypervolume contribution from the sampled subset as the initial item in the selected subset.
    - Remove this item from the sampled subset.
3. **Incremental Selection**:
    - For each iteration, randomly select a small batch (e.g., 1-5% of the remaining sampled items) from the sampled subset.
    - Compute the hypervolume contribution of each item in the batch when added to the current selected subset.
    - Choose the item with the highest hypervolume contribution and add it to the selected subset.
    - Continue this process until a predefined stopping criterion is met (e.g., a fixed subset size or a maximum number of iterations).
4. **Subset Refinement (Optional)**:
    - After obtaining an initial selected subset using the above approach, optionally refine the subset using a more computationally intensive method if needed, such as the greedy algorithm applied to the smaller subset.

By initially sampling a small subset and incrementally selecting items based on their hypervolume contribution, we significantly reduce the computational burden compared to exhaustively evaluating all 500 million items. This approach sacrifices a bit of the subset quality for computational efficiency, but it should still yield a reasonably good approximation of the optimal subset. The exact parameters such as the percentage of items to sample and batch size for incremental selection can be tuned based on computational resources and desired subset quality.

Method 2

The points which are closed to the current selected points are likely to give less contribution, so let's search for a point as far as possible.

The points (x_i,y_i) can be sorted according to x_i value firstly.

If the current solution has k items, we can draw their x_i in such figure:

(x_{left} is x of the most left point and x_{right} is x of the rightest point)

1. For each interval j, we can choose the most middle point suing binary search; then calculate the distance from it to the closest end of the interval d_j. (For the most left interval and the rightest interval, just use the x_{left}, x_{right} instead of the middle point)
2. Then chose the point with largest distance d_j, add it to the current solution.
3. Update the figure. (Remain the other middle points and find new middle points for new interval)

Task2

When dealing with a candidate set size as large as 500 million items, random selection for local improvement can indeed be inefficient. Instead, we can employ more sophisticated techniques to choose a single item to be added to the current subset. One effective approach is to use a combination of local search and heuristic methods. Here's a detailed idea:

1. **Divide and Conquer**:
    - Break down the candidate set into smaller, manageable subsets. This could be done based on some criteria such as clustering similar items together or dividing the space into regions.
    - Apply local search techniques within each subset to identify potential candidate items for addition to the current subset. This can involve methods like hill climbing, simulated annealing, or tabu search.
2. **Heuristic Evaluation**:
    - Develop heuristics to evaluate the potential contribution of each item within the subset to the overall quality of the solution. These heuristics should take into account the specific optimization criteria, such as maximizing hypervolume, while also considering factors like diversity and coverage.
    - Some potential heuristics could include assessing the distance or similarity of the item to the existing subset, its impact on the distribution of points in the objective space, or its potential to fill gaps in the current subset.
3. **Priority Queue**:
    - Maintain a priority queue or heap data structure to efficiently select the most promising candidate item from the evaluated subset. This allows us to prioritize items based on their heuristic scores, ensuring that the most beneficial item is selected for addition to the subset.
4. **Dynamic Adjustment**:
    - Dynamically adjust the search strategy and heuristics based on the characteristics of the candidate set and the current state of the subset. For example, if certain regions of the objective space are underrepresented in the subset, prioritize selecting items from those regions.

By combining local search techniques with heuristic evaluation and efficient data structures, we can effectively choose a single item to be added to the current subset in the local improvement phase. This approach balances computational efficiency with the need to find high-quality solutions in large candidate sets.