Fix n so that we search for all bent functions in $2n$ variables up to linear terms, which we will store in $\mathcal{B}$. Let $\mathcal{S}$ be the set of all semi-bent functions in $2n-1$ variables up to linear terms. Finally, let $\mathcal{R}$ be $RM(1, 2n-1)/\bar{1}$. This is our algorithm in pseudocode:

For f_0 in $\mathcal{S}$ do
    For f_1 in $\mathcal{S}$ do
        If $f_0|_{RM(n,2n-1)} \neq f_1|_{RM(n,2n-1)}$ then
            continue;
        fi;
        For r in $\mathcal{R}$ do
            next = false;
            For u in $\mathbb{F}_2^{2n-1}$ do
                If $W_{f_0}(u) + W_{f_1+r}(u) \bmod 2^{n+1} == 0$ then
                    next = true;
                    break;
                fi;
            od;
            If next == true then
                continue;
            fi;
            $\mathcal{B}$.add$(f_0|(f_1 + r))$;
        od;
    od;
od;

We can cite work on "granted-values" in other papers to prove that f_0 and f_1 must be in $\mathcal{S}$. We know that f_0 and f_1 must agree in all degree n terms, because otherwise concatenation rules show that their concatenation would have a term of degree $n+1$, which we know is not possible for a bent function. We know that we need only consider linear terms in the second half, because if the first half were to have linear terms, we could simply add those to both halves, which is the equivalent of adding a linear term on the outside. Also, we need not consider adding $\bar{1}$ in the second half because that is simply equivalent to adding x_1 on the outside. Finally, we must show that the condition $\forall u \in \mathbb{F}_2^{2n-1}, W_{f_0}(u) + W_{f_1+r}(u) \bmod 2^{n+1} \neq 0$ is both necessary and sufficient. Let $f = f_0|\,(f_1 + r)$. We know that for any $W_f(u)$, $u = u_0|u_0$ or $u = u_0|(u_0 + \bar{1})$. Suppose the first case. Then

$$\begin{aligned} W_f(u) &= W_{f_0|(f_1+r)}\,(u_0|u_0) \\ &= W_{f_0}(u_0) + W_{f_1+r}(u_0) \end{aligned}$$

Clearly, since $W_{f_0}(u_0) = 0$ or $\pm 2^n$ and $W_{f_1+r}(u_0) = 0$ or $\pm 2^n$, $W_f(u) = 0, \pm 2^n$, or $\pm 2^{n+1}$. Since we only want $W_f(u) = \pm 2^n$, the condition $W_{f_0}(u) + W_{f_1+r}(u) \bmod 2^{n+1} \neq 0$ is necessary and sufficient. Now, if $u = u_0|(u_0 + \bar{1})$, you can easily check that we will simply get $W_{f_0}(u_0) - W_{f_1+r}(u_0)$, and the condition will still be both necessary and sufficient. So, our algorithm will work, and will construct all bent functions in $2n$ variables up to linear terms, given all semi-bent functions in $2n-1$ variables up to linear terms.

Now, we are interested in using $RM(n+1,2n)$ to construct all semi-bent functions in $2n+1$ variables. First, the two halves must contain the same degree $n+1$ terms, only the second half needs to contain linear terms, and neither half should contain $\bar{1}$. Also, the function should be bent in both halves or bent in neither half, otherwise it cannot have the correct weight. Given information about "granted-values", let $\mathcal{M} = \{f \in RM(n+1,2n)/RM(1,2n) | \forall u \in \mathbb{F}_2^{2n}, W_f(u) = 0, \pm 2^n, \pm 2^{n+1}\}$. If we let $\mathcal{W} = \{W_{f_0}(u) + W_{f_1+r}(u)\} \cup \{W_{f_0}(u) - W_{f_1+r}(u)\}$, then it must contain exactly 2^{2n} 0's and 2^{2n} $\pm 2^{n+1}$'s. Using an argument similar to that from above, this will be a necessary and sufficient condition; however, it will be more complicated to check than the one above was. Despite this, I propose the following algorithm (in pseudocode), where $\mathcal{R} = RM(1,2n)/\bar{1}$ and $\mathcal{B}$ contains all bent functions in $2n$ variables:

```
For f_0 in M do
    For f_1 in M do
        If f_0|_{RM(n+1,2n)} ≠ f_1|_{RM(n+1,2n)} or ( f_0 ∈ B and f_1 ∉ B ) or ( f_0 ∉ B and f_1 ∈ B ) then
            continue;
        fi;
        For r in R do
            next = false;
            values = [];
            For u in F_2^{2n} do
                If W_{f_0}(u) + W_{f_1+r}(u) mod 2^{n+1} ≠ 0 then
                    next = true;
                    break;
                fi;
                If W_{f_0}(u) - W_{f_1+r}(u) mod 2^{n+1} ≠ 0 then
                    next = true;
                    break;
                fi;
                values.add( W_{f_0}(u) + W_{f_1+r}(u) );
                values.add( W_{f_0}(u) - W_{f_1+r}(u) );
            od;
            If next == true then
                continue;
            fi;
            If values.count( 0 ) ≠ 2^{2n} then
                continue;
            fi;
            B.add(f_0|(f_1 + r));
        od;
    od;
od;
```

I propose another speedup, which basically entails thinking of the added linear term r as performing a permutation of the WHT of f_1. If $r = a_1x_1 + a_2x_2 + \cdots + a_{2n-1}x_{2n-1}$, then $W_{f_1+r}(u) = W_{f_1}(u + a)$, where $a = a_1a_2 \cdots a_{2n-1}$. We can arrange so that this a is in fact the index of r in $\mathcal{R}$.