**Algorithm 3** Swarm(PV P)

1. for j=1 to NH do   
2. count\_zero[j] = 0; // Khởi tạo biến đếm cho các bit 0  
3. count\_one[j] = 0; // Khởi tạo biến đếm cho các bit 1  
4. end for   
5. for i=1 to N do // Duyệt qua toàn bộ quần thể PV   
6. Calculate dis = BitDiff(Pᵢ, P) using Eq. 11; // Tính khoảng cách giữa PV hiện tại và P   
7. if(dis ≤ VD) then // Nếu khoảng cách nằm trong giới hạn VD   
8. for j=1 to NH do // Duyệt qua các bit của PV   
9. count\_zero[j]++ if Pᵢ(j) is zero; // Tăng biến đếm nếu bit thứ j là 0   
10. count\_one[j]++ if Pᵢ(j) is one; // Tăng biến đếm nếu bit thứ j là 1   
11. end for   
12. end if   
13. end for   
14. for j=1 to NH do // Xác định bit của Cen\_AF dựa trên số lượng bit 0 và 1   
15. if(count\_one[j] ≥ count\_zero[j])   
16. Cen\_AF[j] = 1; // Nếu số lượng bit 1 nhiều hơn hoặc bằng, bit thứ j của Cen\_AF sẽ là 1   
17. Else   
18. Cen\_AF[j] = 0; // Ngược lại, bit thứ j của Cen\_AF sẽ là 0   
19. end if   
20. end for   
21. if (f(Cen\_AF) ≥ min\_util AND IS(Cen\_AF) ∉ SHU) then // Nếu Cen\_AF đạt ngưỡng lợi ích tối thiểu và chưa có trong SHU   
22. IS(Cen\_AF) → SHU; // Thêm Cen\_AF vào tập SHU   
23. end if   
24. if (f(P) < f(Cen\_AF)) then // Nếu lợi ích của P thấp hơn Cen\_AF   
25. is\_swarm = true; // Đánh dấu rằng hành vi swarm xảy ra   
26. dis' = BitDiff(Cen\_AF, P); // Tính sự khác biệt giữa Cen\_AF và P   
27. Randomly generate a positive integer k no higher than dis'; // Sinh số ngẫu nhiên k nhỏ hơn hoặc bằng dis'   
28. Update P by applying bitwise complement operation on k bits of it; // Cập nhật P bằng cách đảo k bit khác biệt   
29. if (I(P) ≥ min\_util AND IS(P) ∉ SHU) then // Nếu P đạt ngưỡng tối thiểu và chưa có trong SHU   
30. IS(P) → SHU; // Thêm P vào SHU   
31. end if   
32. end if