```
#include ...
using namespace std;

const int MAXN = 105;
int g[2 * MAXN][2 * MAXN]; // 0 source, n+m+1 sink
int flow[2 * MAXN][2 * MAXN];
int prev[2 * MAXN];
pair <double, double> gophers [MAXN];
pair <double, double> holes [MAXN];


bool canReach(int i, int j, double max_dist){
    double x1 = gophers[i].first; double y1 = gophers[i].second;
    double x2 = holes[j].first; double y2 = holes[j].second;
    double d = sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2));
    if (d <= max_dist) return true;
    return false;
}

int max_flow(int s, int t){
    int max_flow = 0;

    for (int i = s; i <= t; i++)
        for (int j = s; j <= t; j++)
            flow[i][j] = 0;

    while (true){
        // Find path s to t
        for (int i = s; i <= t; i++)
            prev[i] = -1;

        queue <int> q;
        q.push(s);
        prev[s] = -2;
        while (q.size() > 0){
            int u = q.front(); q.pop();
            if (u == t) break;
            for (int v = s; v <= t; v++){
                if (prev[v] == -1 and g[u][v] - flow[u][v] > 0){
                    q.push(v);
                    prev[v] = u;
                }
            }
        }
        if (prev[t] == -1) break;

        // Find bottlneck
        int curr = t;
        int bottleneck = 1 << 30;
        while (curr != s){
            bottleneck = min(bottleneck, g[prev[curr]][curr] - flow[prev[curr]][curr]);
```

```
            curr = prev[curr];
        }

        // Pump
        curr = t;
        while (curr != s){
            flow[prev[curr]][curr] += bottleneck;
            flow[curr][prev[curr]] -= bottleneck;
            curr = prev[curr];
        }

        // Add flow to answer
        max_flow += bottleneck;
    }

    return max_flow;
}


int main(){
    int m, n, sec, vel;
    while (cin >> n >> m >> sec >> vel){
        int s = 0;
        int t = n + m + 1;

        for (int i = 0; i <= t; i++)
            for (int j = 0; j <= t; j++)
                g[i][j] = 0;

        for (int i = 0; i < n; i++){
            double x, y;
            cin >> x >> y;
            gophers[i] = make_pair(x, y);
            g[s][i + 1] = 1;
        }
        for (int i = 0; i < m; i++){
            double x, y;
            cin >> x >> y;
            holes[i] = make_pair(x, y);
            g[i + 1 + n][t] = 1;
        }
        double max_dist = sec * vel;
        for (int i = 0; i < n; i++)
            for (int j = 0; j < m; j++)
                if (canReach(i, j, max_dist)) {
                    g[i + 1][n + 1 + j] = 1;
                }
        cout << n - max_flow(s, t) << endl;
    }
}
```