General approach to optimizing

Basically, when you begin working on a problem, you have no idea where the maximimum or minimum is located. Thus, you should begin with a global optimizer. For example, we could graph the function, or we could use contour plot to get an approximate idea where the max or min is. Both of these are convenient. Neither gives a very accurate answer. Sometimes a graph is too flat to give a good indication of where the max or min is, and in this case the contour plot is likely not to give much useful information. Another global optimizing method is the random search method, which we'll discuss now.

Once you have zeroed in on the area containing the desired max or min, then you can use a local optimizer such as the Newton's method, working in a much smaller area, to find a much more accurate numerical answer. We'll discuss a multivariable version of Newton's method later.

Random Searches for Maxima/Minima. Notes on 3.2

> restart:
r1:=sqrt((x-1)^2+(y-5)^2)^0.91:
r2:=sqrt((x-3)^2+(y-5)^2)^0.91:
r3:=sqrt((x-5)^2+(y-5)^2)^0.91:
r4:=sqrt((x-1)^2+(y-3)^2)^0.91:
r5:=sqrt((x-3)^2+(y-3)^2)^0.91:
r6:=sqrt((x-5)^2+(y-3)^2)^0.91:
r7:=sqrt((x-1)^2+(y-1)^2)^0.91:
r8:=sqrt((x-3)^2+(y-1)^2)^0.91:
r9:=sqrt((x-5)^2+(y-1)^2)^0.91:
F:=3.2+1.7*(6*r1+8*r2+8*r3+21*r4+6*r5+3*r6+18*r7+8*r8+6*r9)/84:

This function, involving the distances from (x,y) to 9 different points, each raised to the 0.91 power, is a model for the average response time of emergency crews for if a new fire station is placed at location (x,y) in a city. Note that this function has many places where it is not differentiable. Calculus methods are not going to help here.

Because there are only two variables, we can use the graph and level curves to get a rough idea where the optimal location is.
> plot3d(F, x=0..6, y=0..6, axes=boxed);
> with(plots): contourplot(F, x=0..6, y=0..6, axes=boxed, color=black, grid=[40,40], contours=10);
For what it is worth, we can ask Maple to set the derivatives equal to zero, too. This may not be very reliable, since the derivatives are not even defined everywhere.

\begin{align}
F1 &:= \text{diff}(F, x); F2 := \text{diff}(F, y); \\
F2 &:= \frac{0.0552499999 (2 y - 10)}{(x^2 - 2 x + 26 + y^2 - 10 y)^{0.5450000000}} + \frac{0.0736666666 (2 y - 10)}{(x^2 - 6 x + 34 + y^2 - 10 y)^{0.5450000000}} \\
&\quad + \frac{0.0736666666 (2 y - 10)}{(x^2 - 10 x + 50 + y^2 - 10 y)^{0.5450000000}} + \frac{0.1933750000 (2 y - 6)}{(x^2 - 2 x + 10 + y^2 - 6 y)^{0.5450000000}} \\
&\quad + \frac{0.0552499999 (2 y - 6)}{(x^2 - 6 x + 18 + y^2 - 6 y)^{0.5450000000}} + \frac{0.0276250000 (2 y - 6)}{(x^2 - 10 x + 34 + y^2 - 6 y)^{0.5450000000}} \\
&\quad + \frac{0.1657500000 (2 y - 2)}{(x^2 - 2 x + 2 + y^2 - 2 y)^{0.5450000000}} + \frac{0.0736666666 (2 y - 2)}{(x^2 - 6 x + 10 + y^2 - 2 y)^{0.5450000000}} \\
&\quad + \frac{0.0552499999 (2 y - 2)}{(x^2 - 10 x + 26 + y^2 - 2 y)^{0.5450000000}} \\
\end{align}

\>( s := \text{fsolve}([F1=0, F2=0], [x, y], x=0..6, y=0..6); \\
\text{s := \{x = 1.616487858, y = 2.765906920\}} \\
\>( \text{assign(s); F}; \\
\text{6.462980432} \)
Here is a random search algorithm, which randomly picks 1000 points in the region, compares the value of F at each new point to the optimal one found so far, and keeps track of the location and value of the optimal one found in the process.

Notice that the Maple command `rand()`, each time it is called, randomly picks a 12 digit number. Thus \( \frac{\text{rand()}}{10^{12}} \) provides us, each time it is called, a number between 0 and 1. For more info on the random number generator `rand` type `? rand`. In the algorithm below, we multiply the random number between 0 and 1 by the length of the interval, then add it to the left endpoint, to get a random point in the interval.

For this example, the region in which we want to optimize F is \( 0 \leq x \leq 6, 0 \leq y \leq 6 \).

```maple
> a:=0:b:=6:c:=0:d:=6:N:=1000:
x:=evalf(a+(b-a)*rand()/10^12):
y:=evalf(c+(d-c)*rand()/10^12):
zmin:=F:
for n from 1 to N do
    x:=evalf(a+(b-a)*rand()/10^12):
y:=evalf(c+(d-c)*rand()/10^12):
z:=F:
    if z<zmin then
        xmin:=x:
ymin:=y:
zmin:=z:
    fi:
od:
xmin;ymin;zmin;
```

```
1.571826343
2.732162340
6.463947540
```

Repeated executions of this random search program give different answers, fairly close to the approximate optimum found by `fsolve`.

```maple
> unassign('x'): unassign('y');
> s:=fsolve({F1=0,F2=0},{x,y},x=1.4..1.7,y=2.5..3);
s := \{x = 1.616487858, y = 2.765906920\}
```

```maple
> assign(s); F;
6.462980432
```