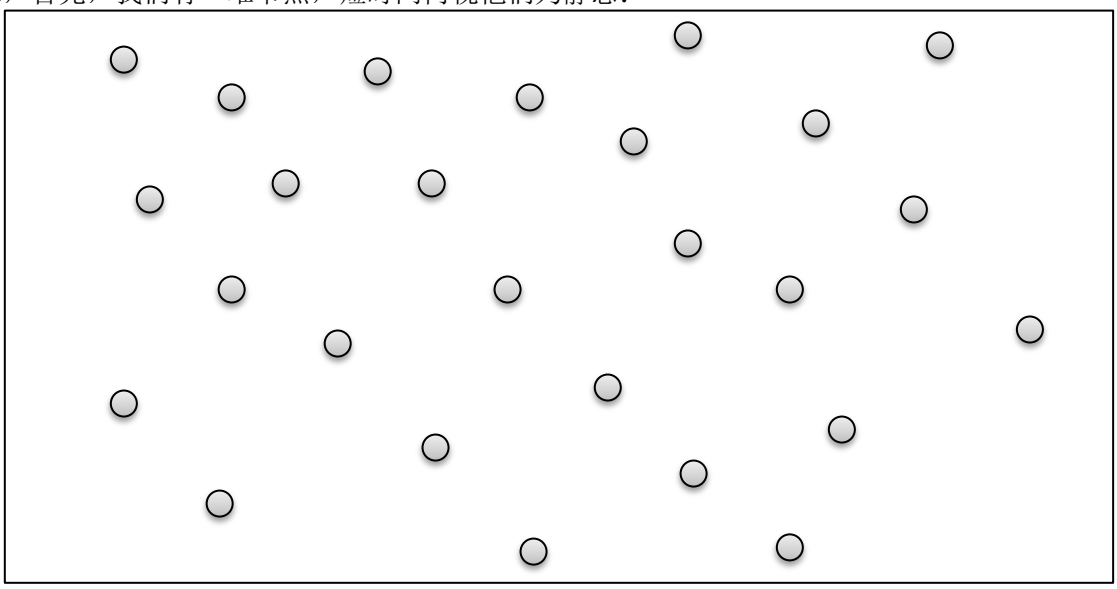
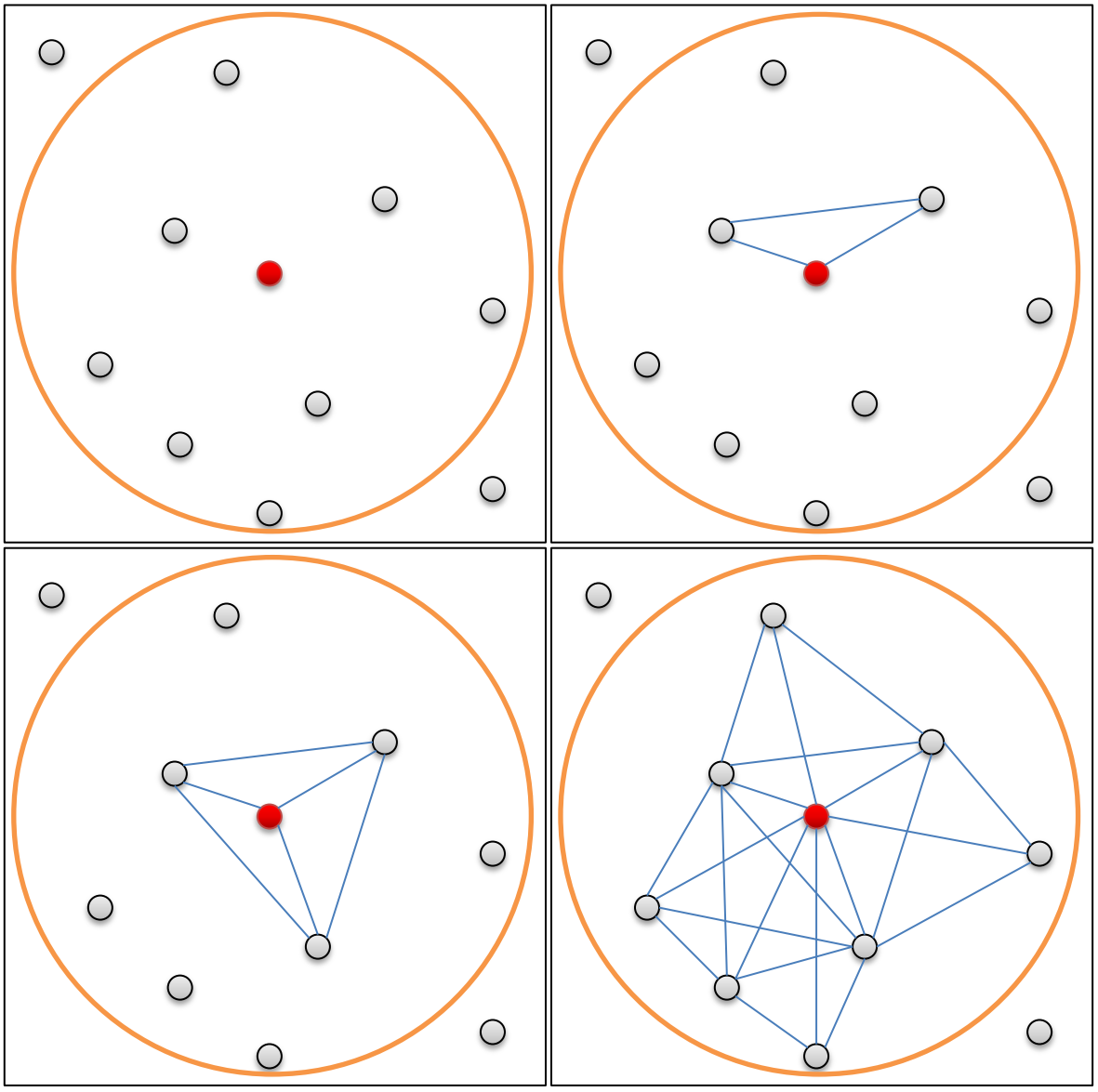
嗯，首先，我们有一堆节点，短时间内视他们为静态：

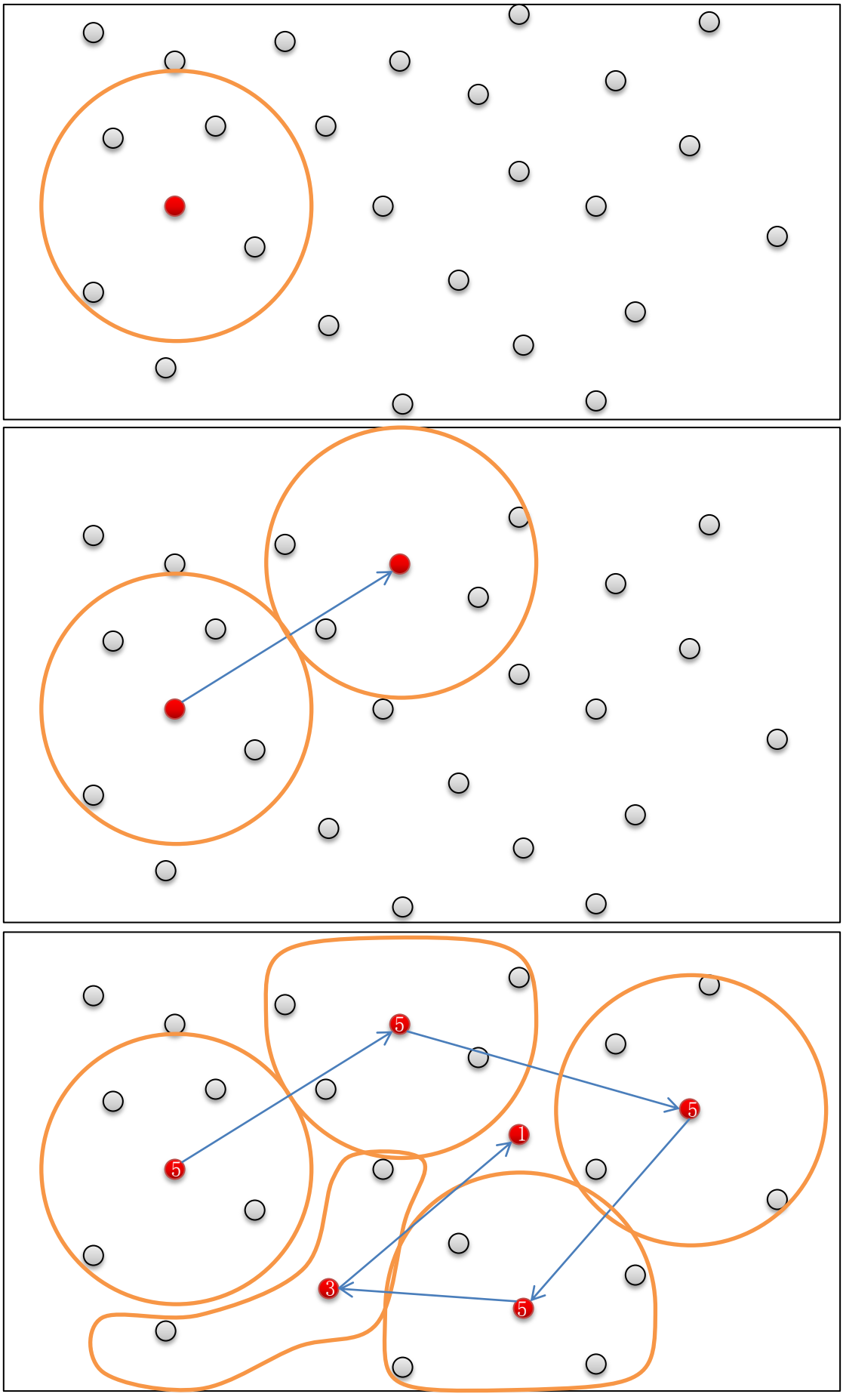


我们需要对它们进行定位，同时希望定位及决策的速度快一些，于是决定选一些 leader 出来，每个 leader 节点会组织离自己一定范围内的节点进行互相定位（由近及远）：



每个节点的定位至少由另外 3 个节点完成（第一个三角形除外），这样可以保证可以定出正确的方位。

然而上位机可能无法完全感知到所有节点的存在，我们需要随机指定一个初始的 leader 节点，并由它指定下一个 leader 节点：



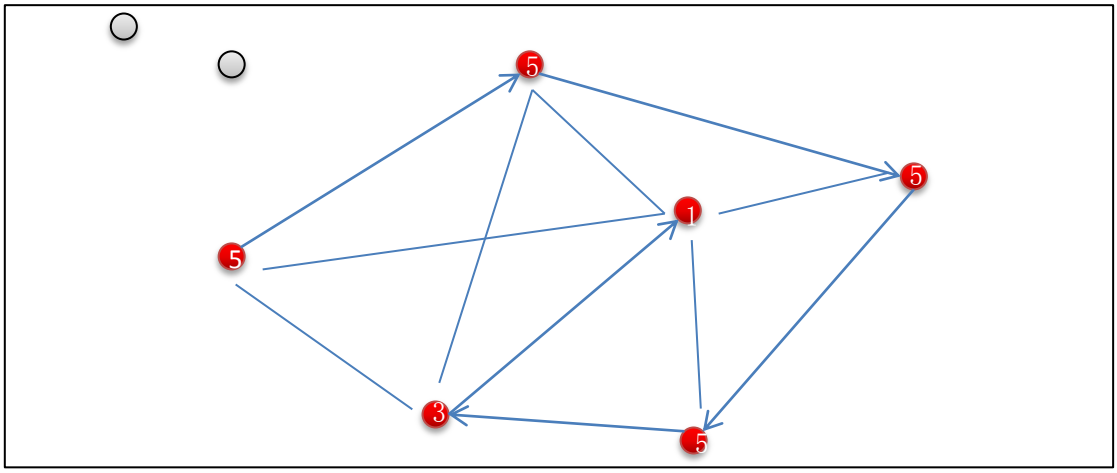
以实际情况决定每个 leader 节点所管辖的范围半径（可以由上位机指定，按实际应用需求确定）
以上步骤的具体实现：

1、 leader 节点广播
2、 各节点反馈（可以附加上随机的时间增量），测量距离
3、 leader 节点选定需要的子节点
4、 leader 节点选定下一任 leader 节点，最理想的下一任应该是 2 倍半径处的节点
5、 广播这些决定，属于它的子节点转变接收模式，不再响应别的 leader 节点的号召
6、 此 leader 节点组织子节点进行互相定位
7、 下一任 leader 节点重复此步骤

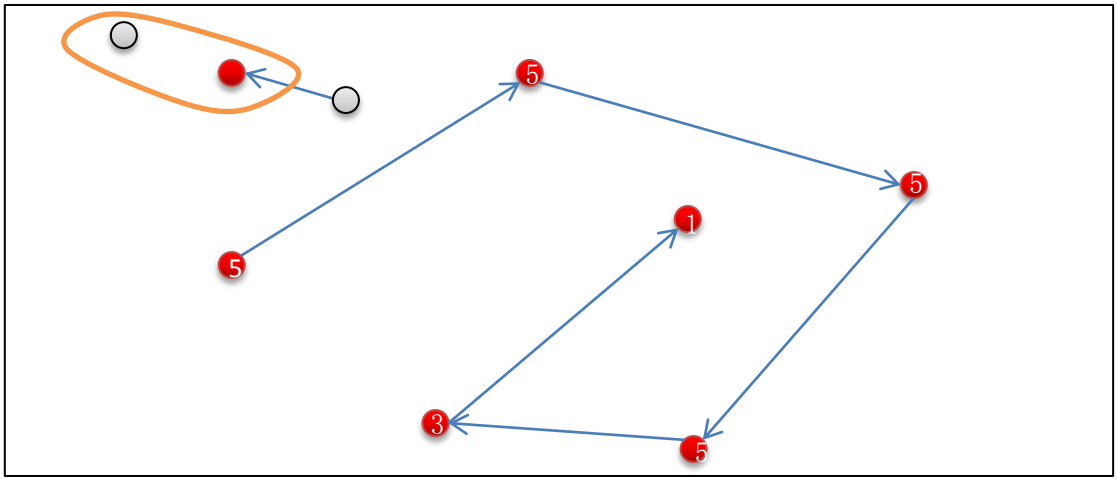
优势：

在节点较多的系统中，此方案可以减少大量的时间，因为此时可以不再同一时间只有一个节点参与定位，而是多个子系统同时在内部进行定位，得出部分节点的相对位置。这样时间主要都花费在 leader 节点的流转中，较全部节点逐一定位要快速得多。

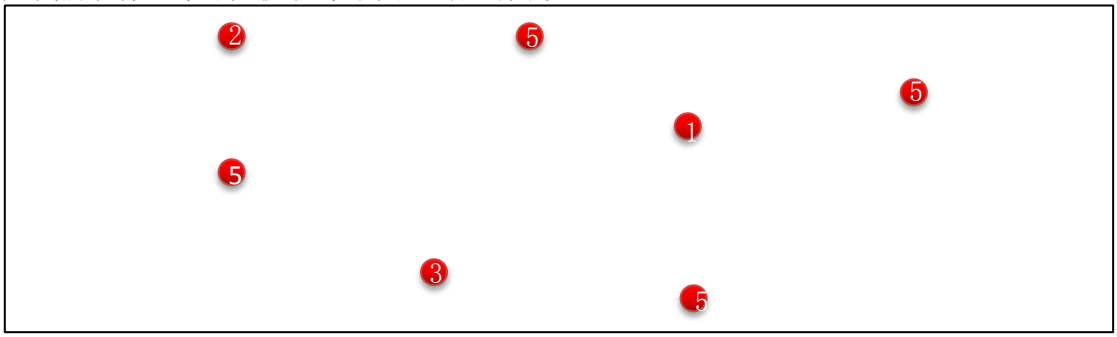
当前面的 leader 节点已经完成了内部定位之后，可以互相之间进行定位，以确定每个小图形的相对位置：



于是我们知道了图形的边界。也许有某些节点没有响应最后的 leader 节点的探测（如距离太远），我们用边界节点（当然也可以只用临近边界的 leader 节点）再探测一次：

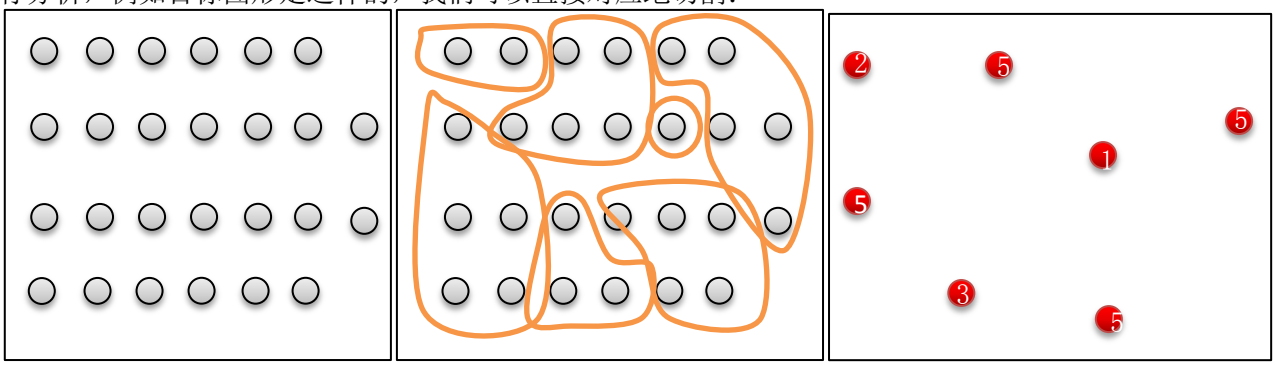


然后我们就有全部可以联系到的节点组成的图了：



这时，我们需要对整个图的方向进行定位，例如我们需要知道哪边是北。我们可以让任意数个节点做已知方向的相对运动（利用电子罗盘等工具），那么它们相对于整体的运动方向就可以确定，于是整体的方向就出来了，当前的分布情况定位就基本完成了。

当我们需要组建新的图形时，使用 leader 节点的另一好处就体现了出来，可以直接据此对目标图形进行分析，例如目标图形是这样的，我们可以直接对应地切割：

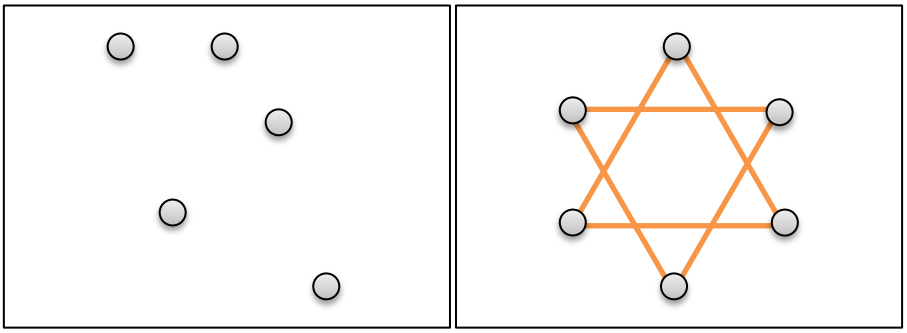


然后分配给每个 leader，每个 leader 再对子节点进行调度，这样如果算法复杂度较高，可以很好地节省时间（类似于分治法）。这里使用的算法可以抽象出来专门进行优化研究（与 leader 将分配到的子图与子节点对应的算法类似）。算法的执行者可以是上位机，也可以是任何一个 leader 节点。

至此，整个算法已经基本完成。

最后，如果实际统计到的节点数目与给定的总图不同，我们需要上位机做出决策进行修改，因为大方针的指定只能由一个个体完成，协同合作极有可能出现混乱：

假如我们的当前状态与目标分别为：



那么我们需要决策：改成五角星呢，还是使用六芒星的其中 5 个顶点呢？而此项工作只能由某一个个体决定，指挥中心（上位机）将会是合适的选择。