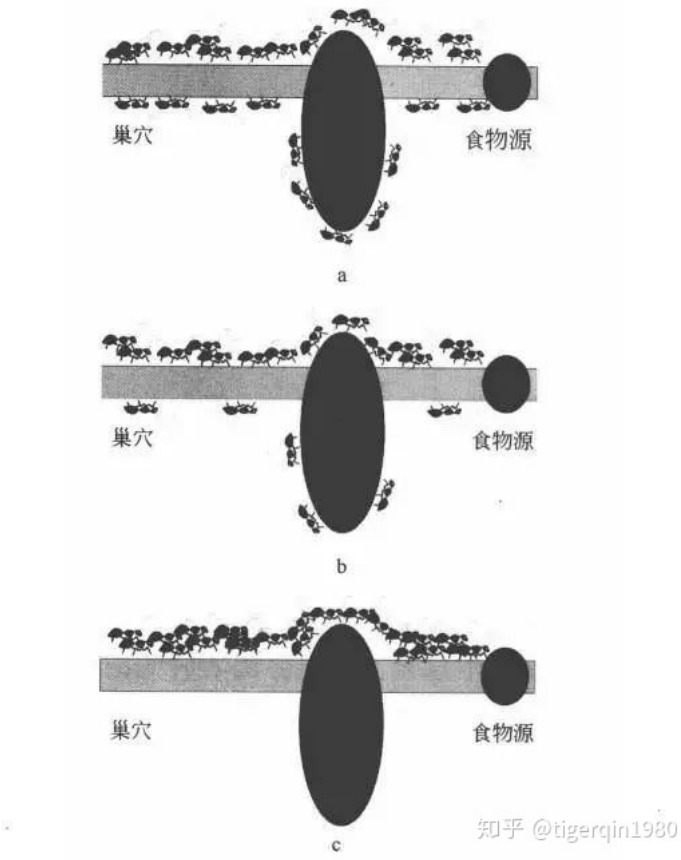
最短路径到达食物源了。



由上述蚂蚁找食物模式演变来的算法，即是蚁群算法。这种算法具有分布计算、信息正反馈和启发式搜索的特征，本质上是进化算法中的一种启发式全局优化算法。 最近几年，该算法在网络路由中的应用受到越来越多学者的关注，并提出了一些新的基于蚂蚁算法的路由算法。同传统的路由算法相比较，该算法在网络路由中具有信息分布式性、动态性、随机性和异步性等特点，而这些特点正好能满足网络路由的需要。

蚁群算法演练

蚁群算法应用广泛，如旅行商问题(traveling salesman problem,简称TSP)、指派问题、Job-shop调度问题、车辆路径问题（vehicle routing problem）、图着色问题(graph coloring problem)和网络路由问题（network routing problem）等等，下面我们同之前推文一样，以TSP的求解为例演练蚁群算法的应用。

关于TSP问题，如果还有疑问，请参考之前的推文: “

干货|十分钟教你用动态规划算法解Travelling Salesman Problem（TSP）问题，附代码……”。

关于求解TSP的蚁群算法可以参考文章: Ant colony system: a cooperative learning approach to the traveling salesman problem,M. Dorigo, L.M. Gambardella, IEEE Transactions on Evolutionary Computation,Volume: 1, Issue: 1, Apr 1997, pages 53 - 66。

1. TSP建模

首先考虑 TSP，给定图 $G(V,E)$, 其中 V 为点集，E 为边集，假设 V 中编号为 1……N，令 c_{ij} 为城市 i 到城市 j 的距离，其中 $i \neq j$，假设 $c_{ii} = M$（M 是一个大数，和图网中距离有关，保证这个 M 至少要比原图中最长的边更长）。这样保证了没有自环。同时令 x_{ij} 为 $0-1$ 变量满足如下约束：

$$x_ij = \begin{cases} 1 & if\ the\ solution\ to\ TSP\ goes\ from\ city\ i\ to\ city\ j \\ 0 & otherwise \end{cases}$$

这样，我们可以用如下模型来描述 TSP：

$$minz = \sum_i \sum_j c_{ij} x_{ij}$$

s.t.

$$\sum_{i=1}^{i=N} x_{ij} = 1 \qquad \forall j \in V$$

$$\sum_{j=1}^{j=N} x_{ij} = 1 \qquad \forall i \in V$$

$$u_i - u_j + N x_{ij} \leq N - 1 \qquad (for i \neq j; \forall i \in V, \forall j \in V, i \neq 1, j \neq 1,)$$

$$\forall x_{ij} = 0\ or\ 1$$

$$\forall u_j \geq 0$$

于 TSP，如果还有疑问，请参考本公众号的推文：“干货 | 十分钟教你用动态规划算法解 Travelling Salesman Problem(TSP) 问题，附代码……”。

知乎 @tigerqin1986

2. 蚁群算法

其经过的路径上释放一种称之为“信息素”的物质，蚁群内的蚂蚁对“信息素”具有感知能力，“信息素”浓度越高的路径就有越大的概率被蚂蚁选择，而每只路过的蚂蚁都会在路上留下“信息素”，这就形成一种类似正反馈的机制，经过一段时间后，最终整个蚁群就会沿着最短路径到达食物源了。因此，我们可以设计类似于蚂蚁找食物的算法来求解 TSP，这样求解出来的解质量不错，但是不能保证是最优解。

我们令 η 为一个启发值 (herustic value)，这个值依赖于边 (r,s)，在 TSP 中这个值通常设定为城市 r 与城市 s 之间距离的倒数。另外设定 为信息素浓度，这个值同样是和边相关。它的初始值我们设定为 $\frac{1}{|V|\cdot Mean}$，其中 $|V|$ 为点的个数，$Mean$ 为 E 中所有边的平均长度。当然，也可以有其它的设定方式，但是要保证量级处于合适的位置。对于蚁群算法，基本的框架就是对于 m 个 ant 或者 agent，让它们分别去构造路径，并留下信息素，在下一次迭代的过程中会利用这些信息素。对于每个 agent，当它需要访问下一个城市的时候，我们让它以一个特定的概率分布对接下来要访问的城市随机进行选择，概率分布如下：

$$p_{rs}^{k}=\begin{cases}\dfrac{\tau_{rs}^{\delta}\cdot\eta_{rs}^{\beta}}{\sum\limits_{z\in J_k(r)}\tau_{rz}^{\delta}\cdot\eta_{rz}^{\beta}} & if s\in J_k(r)\\ 0 & otherwise\end{cases}$$

其中 p_{rs}^{k} 表示 agent k 在 city r 选择 city s 的概率，$J_k(r)$ 表示 agent k 在 city r 还需要访问的城市的集合。δ 和 β 分别表示 τ 和 η 的比重，δ 越大，τ 比重越大，反之亦然。通常设定 $\delta=1$，$beta=6$.

根据上述准则可以构造所有 agent 的路径，当所有 agent 完成一次 tour 以后，更新信息素。更新规则如下：

$$\tau_{rs}\leftarrow(1-\rho)\cdot\tau_{rs}\qquad\forall(r,s)\in E$$
$$\tau_{rs}\leftarrow\tau_{rs}+\Delta\tau_{rs}\qquad\forall(r,s)\in Tour^{ib}$$

其中 E 为边集，ib 为当前迭代最优的 agent 的编号，$Tour^{ib}$ 表示当前迭代最优解集合。$\Delta\tau_{rs}^{ib}$ 是 ib 在边 (r,s) 留下的信息素。$\Delta\tau_{rs}^{ib}$ 的计算方式如下：

$$\Delta\tau_{rs}^{ib}\leftarrow\frac{1}{L^{ib}}$$

其中 L^{ib} 是 $Tour^{ib}$ 的路径长度。

我们设计类似于蚂蚁找食物的算法来求解 TSP，这样求解出来的解质量不错，但是不能保证是最优解。这样经过多次迭代更新后，可以给“蚂蚁”一点点智能和经验，从而得到一个不错的解。

知乎 @tigerqin1980

附. 蚁群算法相关代码

先放上一波严肃的伪代码分析：

▲ 赞同 280 ▼ 12 条评论 分享 喜欢 收藏 申请转载 ···