**运行时预测**：本节我们对上述分类的算法在运行时间与消息到达速率方面进行预测，并展示预测结果。

我们将特征向量表示成X=[x0,x2,…,xm]，即m+1维向量，对每个特征向量X我们都有一个运行时间t，因此点对d(X,t)代表一个样本，整个数据集合为D=[d1,d2,d3,…,dn]。该特征提取于coordinator节点所收集的运行时信息。下面我们使用随机森林、岭回归、局部加权回归、神经网络等方法，分别对上述算法进行训练预测，我们采用10交叉验证的方法，使用均方相对误差做为损失函数，公式如下：

公式

我们之所以采用均方相对误差做为损失函数，是因为相对于均方绝对误差与均方误差而言，均方相对误通过将error/true running time来降低慢任务的影响。例如，现有三个真实running time分别是20ms，50ms，120ms，预测的后的时间对应为15ms，40ms, 100ms。那么如果采用均方误差来衡量模型的性能，得到的结果分别是25，100，400，整个模型的MSE为100，可以看出，120ms的贡献了误差的绝大部份，相反，如果我们使用均方相对误差，得到的结果分别是a,b,c，数据之间的差别很小，因此我们可以更好的捕捉短时间的训练样本。

**第一类算法运行时间预测**：

特征提取：我们从上节分析可知，该类算法worker window size恒定，永远等于vertex number，不会受接收的消息所影响，因此针对此类算法，我们更多的针对子图本分进行特征提取，包括图的拓扑结构，提取的特征如下

Ivnum、tvnum、ovnum、edge\_num、msg\_num、密度、fid (后续可以加入一些embeding的思想)

上述特征均可从IncEval每轮计算中直接得到，因为图中点数目与边数目等信息很难表示图的拓扑结构，因此我们使用中fid来表示，以达到线上训练的有效性与效率性，同样也可以使用embeding等方法将图的拓扑信息表示成向量（这种方式下适合离线训练）。

是否在以某个算法为例，介绍一下过程。

表

从上表中我们可以看出，（那个模型的最小），所有算法的MSRE都非常小，

**第二类算法运行时间预测：**

**特征提取：**与第一类算法不同，该类算法worker window并不恒定，且受接收的消息所影响，因此除包含第一类算法所提取的特征外，我们还需对接收的消息做映射。提取的特征如下：

Msg\_embedding

其中msg\_embedding 是包含消息映射的向量，表示成Z=[z1,z2…,zn]，n为超参数，代表向量的大小，zi代表接收消息的点id（术语）所映射的区间，不同的点映射到相同的区间，则采用累加的方式进行处理，映射的公式如下I=fai/L，其中L为参数代表区间大小,如果L很大，则会提高模型的预测精度，但会增加模型训练的时间，相反，减小L则会极大缩短模型的训练时间，但会降低模型的预测精度。例如，我们现在有100个点…

表

从上表中我们可以看出

**第二类算法运行时间预测：**

**特征提取：**与前两类算法不同的是，该类算法属于多阶段类型，且某一阶段或属于第一种类型或属于第二种类型，但上节分析得出，即便处于多阶段类型，但若干个阶段之间有相似的重复的规律，因此除上述提起的特征外，我们提取处理后轮数做为特征：

表

从上表中我们可以看出

**消息到达速率的预测**：我们采用“最近的平均预测不远的将来”思想来预测消息到达速率，即我们计算t时间内所收集的消息数量m，并用m/t来表示下一轮的消息达到速率。之所以使用这种方式，主要原因如下：1）我们发现，某个fragment接收消息的速率，不仅跟与之相连的其他fragment的个数有关，还与其他fragment中前一轮计算的活跃点结合相关（worker\_window），甚至还需当时网络延迟等多方面不可控因素相关，因此无法采用机器学习的方式提取特征，从而达到想训练时间那样训练消息到达速率，2）我们发现，MPAP模型下消息的到达速率随之t的减小为趋于连续平稳变化，几乎没有凸起值，因此这种最近的平均预测不久的将来思想确实有效，同样我们也用实验的方式验证了其有效性3）即便有很多已有的模型如RNN模型，可以很精确的预测，但这些模型有很昂贵的部署代价，相反，我们的思想代价极低。

**实验（系统的设计与实现做为中的一个小节）**

我们在GRAPE+的基础上开发了支持MPAP模型的GRAPE++。

GRAPE+的架构如图所示，其中最上层用户结构层仍然采用与GRAPE+相同的架构，即用户需指定注册PIE函数，中间层是GRAPE+的核心部分，包括

GRAPE+在以下方面与之不同

1. runtime info collector
2. MPAP scheduler
3. Modle train （RPC）