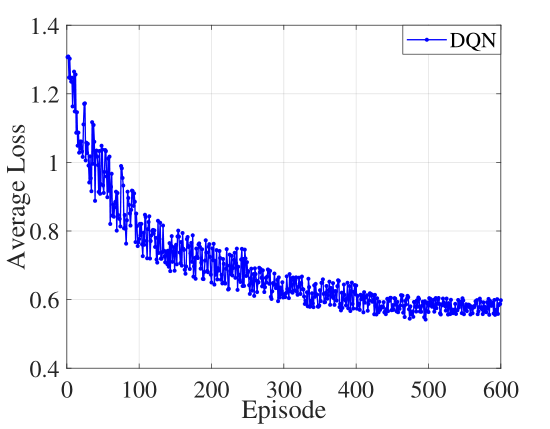
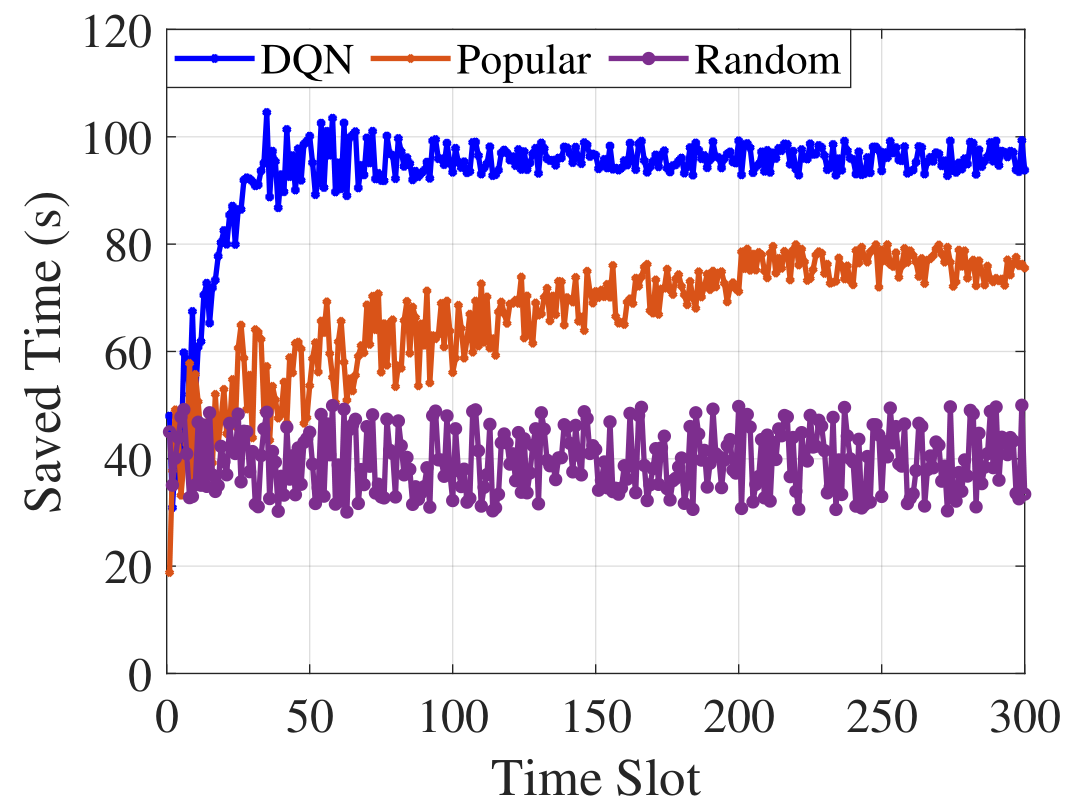
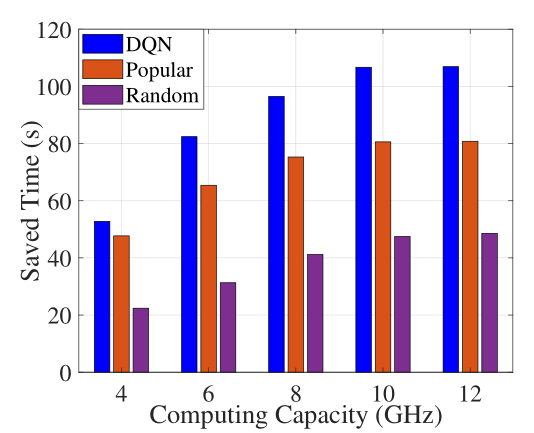
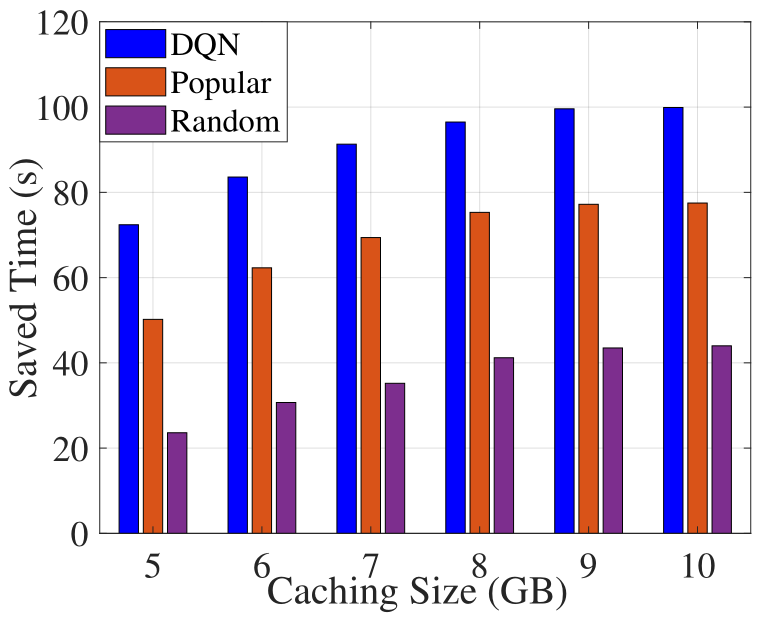
实验环境：共有10个服务内容，边缘基站的计算能力为8GHz，缓存空间为8GHz（前两张图的实验条件），每个服务内容需要的计算量为[1GHz,4GHz]之间，需要的缓存空间为[2GB,4GB]之间。服务的流行度符合Zipf分布，用户请求的到达符合泊松分布。



该图显示了算法在训练过程时Average loss的变化过程，体现了算法能够有效收敛。在前450次左右的训练中Average loss 逐渐下降且下降幅度逐渐减小，在450次后算法已趋于平稳。另外，随着Episode的增加，算法上下震动的幅度也可以看到明显减小。



该图显示了算法在性能（即通过边缘计算节约的时间）方面的表现，与两种算法进行了对比，Popular是指在满足边缘基站计算能力和缓存空间约束的前提下，尽可能将流行度高的服务进行边缘基站本地计算。Random是指在满足上述两个约束下随机选择服务进行边缘基站本地计算。其中可以看到，所提DQN算法具有最优的性能，且其收敛速度比Popular算法更快，这是因为DQN模型可以先通过离线的方式进行训练，因此在在线运行时就可以快速收敛。但Popular算法由于需要通过根据用户请求等信息逐步计算流行度，而当前期信息较少时计算出的流行度会变化很大，因此收敛速度较慢。随机算法由于每次随机选择服务，因此波动幅度较大。

这两张图表现了三种算法在不同约束条件下的性能变化趋势。左图是不同计算能力下三种算法节约时间的变化，可以看到随着边缘基站计算能力的增强，算法节约时间也随之增加，但这种增加趋势不会一直进行下去，如图中当计算能力在10GHz和12GHz时算法节约时间基本一致，这是因为此时边缘基站的计算能力不再是算法性能的瓶颈，而是受到边缘基站缓存空间的限制，无法缓存更多的服务，从而导致算法性能无法进一步提升。另外，从图中可以看出，在计算能力较小时Popular算法性能提升较快，而随着计算能力增加，提升幅度迅速减小，如在4GHz到6GHz过程中性能幅度提升较大，而在8GHz到10GHz过程中幅度减小。这是因为Popular算法随计算能力增加后加入了一些流行度相对较低的服务，所以幅度相对减小，而所提DQN算法由于可以根据约束条件主动变换调整边缘计算策略，如随着计算能力增加，缓存空间逐渐成为约束时，将原某些计算需求小缓存大的服务更换为计算需求大缓存小的服务，从而使得算法性能稳定上升。另外，由于随机算法的随机性，当缓存空间成为瓶颈前，其性能基本随计算能力的增加而线性增加。右边图情况类似。综合两张图可以看出，单个约束条件对算法性能的影响会逐渐减小，只有同时增加两方面能力才能实现算法性能持续增加。