的设计[10].

(1) 染色体编码

在进行遗传操作之前, 首先需要对优化问题的可行解进行编码. 常用的编码方法有二进制编码、格雷码编码和实数编码等, 其中二进制编码存在汉明悬崖问题, 格雷码编码在进行交叉操作时会产生更高阶的非线性. 为了克服这些问题, 这里采用实数编码方案. 对于问题 (6) 的一个可行解$(n_1,n_2,\cdots,n_N)$, $n_i \in N_0$且$0 \leqslant n_i \leqslant n_i^{\max}$, 其中$n_i^{\max} = \lceil D_{\max}/C_i \rceil$, $D_{\max}$为工作负载需求的最大值, n_i可以用一个实数变量x_i表示$0 \leqslant x_i$. 因此, 可行解$< n_i^{\max}+1(n_1,n_2,\cdots,n_N)$可以编码为$(x_1,x_2,\cdots,x_N)$.

(2) 适应度函数

适应度函数用于评价种群中个体对环境的适应程度, 也就是可行解的优劣, 可行解越好, 适应度函数值越大. 问题 (6) 是一个最小化问题, 以总的资源提供成本最小化为优化目标. 对于可行解$(n_1,n_2,\cdots,n_N)$, 适应度函数$f(n_1,\cdots,n_N)$定义为

$$f(n_1,\cdots,n_N) = \frac{1}{1+T(n_1,\cdots,n_N)-T_{\min}} \tag{8}$$

其中, $T(n_1,\cdots,n_N)$是虚拟机预留配置为$(n_1,\cdots,n_N)$时总的资源提供成本, $T_{\min}$是到目前为止所获得的目标函数的最小值.为了确定$T(n_1,\cdots,n_N)$,需要求解N_S个子问题 (7).

(3) 选择

选择算子实现种群的优胜劣汰, 既要保证一定的选择强度, 使得种群向着最优解进化, 又要保持种群的多样性, 防止未成熟收敛. 常用的选择算子有适应度比例选择、锦标赛选择和排序选择等. 根据 Blickle 等人[11]的研究, 指数排序选择算子能够在相同的选择强度条件下, 保证较好的种群多样性, 避免算法陷入局部最优解, 因此本文采用指数排序选择算子. 在指数排序选择操作中, 首先对种群中所有个体按照适应度值升序排列, 然后按照个体的顺序为其分配相应的选择概率

$$p_i = \frac{c-1}{c^N-1}c^{N-i} \tag{9}$$

其中, $c = p_i/p_{i+1}(0<c<1)$是相邻两个个体选择概率的比值, c值越小, 选择算子的选择强度越大. 在确定了每个个体的选择概率之后, 使用轮盘赌采样法选择个体.

(4) 交叉

交叉操作是遗传算法的关键步骤, 决定了遗传算法的全局搜索能力. 本文采用模拟二进制交叉算子 (SBX)[12], 对于大多数优化问题具有良好的全局搜索能力. 在 SBX 操作中, 对于两个父个体$\beta_i = (\beta_{i1},\beta_{i2},\cdots,\beta_{iN})$和$\beta_j = (\beta_{j1},\beta_{j2},\cdots,\beta_{jN})$, 两个子个体$\beta_i'$和$\beta_j'$定义为

$$\begin{cases} \beta'_{ik} = \dfrac{1}{2}[(1+B_k)\beta_{ik} + (1-B_k)\beta_{jk}] \\ \beta'_{jk} = \dfrac{1}{2}[(1-B_k)\beta_{ik} + (1+B_k)\beta_{jk}] \end{cases} \tag{10}$$

其中, B_k定义为

$$B_k = \begin{cases} (2\cdot u)^{\frac{1}{\eta+1}}, & u \leqslant \frac{1}{2} \\ \left(\frac{1}{2\cdot(1-u)}\right)^{\frac{1}{\eta+1}}, & u > \frac{1}{2} \end{cases}$$

其中, u是[0,1]上均匀分布的随机数, η取值为 2.

(5) 变异

变异算子决定了遗传算法的局部搜索能力, 同时也维持了种群的多样性. 常用的变异算子有均匀变异、高斯变异和 Breeder 变异等. 均匀变异和高斯变异的性能依赖于参数的自适应调整, Breeder 变异的性能稍差, 但是更容易实施. 在 Breeder 变异操作中,

$$n_i' = n_i \pm (0.5\cdot range_i\cdot\theta) \tag{11}$$

其中, 正负号以相等的概率随机选择, $range_i$是n_i的取值范围, θ定义为

$$\theta = \sum_{m=1}^{M} a(m)\cdot 2^{-(m-1)}$$

其中, $a(m)$以概率$1/M$取值为 1, 以概率$1-1/M$取值为 0.

遗传算法的流程为:

参数设置: 在遗传算法中, 种群规模N、交叉概率p_c和变异概率p_m与算法的性能密切相关. 通常: 种群规模设置为大于变量数的 10 倍, 交叉概率设置为 0.7–0.8 之间, 变异概率设置为变量数的倒数[10].

步骤 1. 随机生成问题 (6) 的N个可行解作为初始种群;

步骤 2. 使用指数排序选择法选择当前种群中的个体进行复制, 执行N次, 生成N个个体;

步骤 3. 将选择-复制操作生成的N个个体进行随机配对, 对每一对个体, 以交叉概率p_c执行 SBX 操作;

步骤 4. 对交叉操作生成的每一个个体, 以变异概率p_m执行 Breeder 变异操作;

步骤 5. 判断是否满足终止条件, 若满足则终止遗传算法, 否则返回步骤 2 继续执行遗传算法.

3 仿真实验

使用某网站一个月的工作负载历史数据作为实验数据, 如图 1 所示. 根据工作负载历史数据, 可以对工作负载的概率分布进行估计, 结果如图 2 所示. 针对 Web 服务, 云提供商 (Amazon EC2) 提供了 4 种类型的虚拟机: Small(m1.small), Medium(m1.medium), Large(m1.large), Extra Large(m1.xlarge), 它们的计费策略和服务能力[7]如表 1 所示.

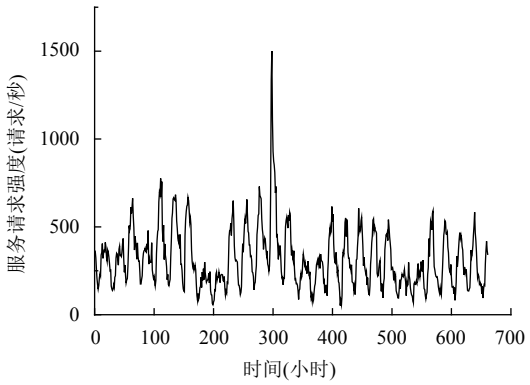

图 1 工作负载

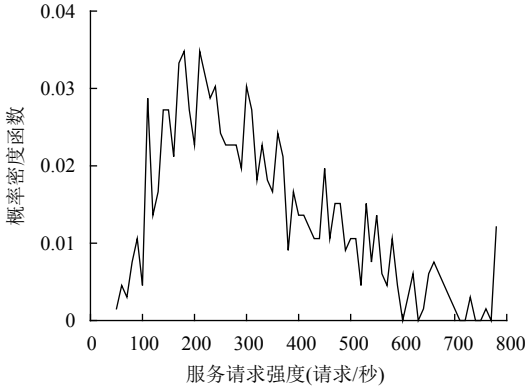

图 2 工作负载的概率分布

表 1 每种虚拟机类型的计费策略和服务能力

类型	p_i^R(美元/小时)	p_i^r(美元/小时)	p_i^o(美元/小时)	C_i(请求/秒)
Small	0.014	0.010	0.044	10
Medium	0.028	0.020	0.087	30
Large	0.056	0.042	0.175	65
Extra Large	0.113	0.083	0.35	100

3.1 资源预留方案对运营成本的影响

首先分析资源预留对云消费者的资源使用成本的影响. 研究不同的资源预留方案下云消费者的资源使用成本, 结果如图 3 所示. 从图中可以看出, 最优的资源预留方案为$\{n_1=0, n_2=0, n_3=4, n_4=0\}$, 预留的服务容量为 260 请求/秒, 资源使用成本为 5694 美元. 随着预留资源的增加, 预留实例的使用成本逐渐增大, 按需实例的使用成本逐渐减小, 在预留资源太多或者太少时资源使用成本均比较高, 这是因为在预留资源太多时, 预留实例得不到充分使用, 造成资源浪费, 在预留资源太少时, 需要使用更多的按需实例.

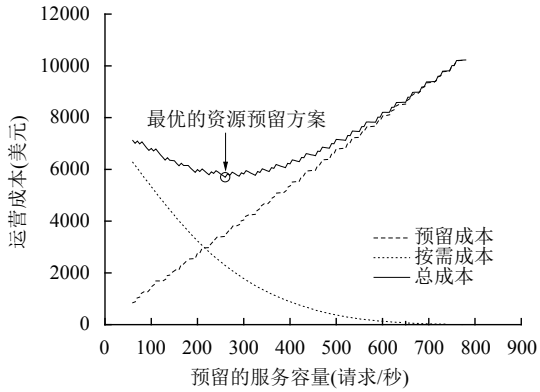

图 3 资源预留方案对运营成本的影响

3.2 抽样平均近似方法

接下来分析抽样平均近似方法的性能, 比较均匀离散化方法与蒙特卡罗方法及拟蒙特卡罗方法的近似精度, 并讨论抽样点数对近似精度的影响, 结果如图 4 所示. 从图中可以看出, 抽样点数越大, 抽样平均近似方法的近似精度越高, 在相同的抽样点数下, 均匀离散化方法比蒙特卡罗方法及拟蒙特卡罗方法的近似精度更高, 当抽样点数$N_S=10$时, 均匀离散化方法的近似精度达到 99%. 因此, 选择均匀离散化方法, 并且选取抽样点数$N_S=10$.

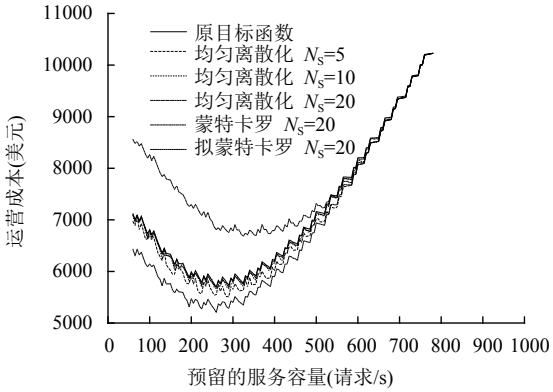

图 4 抽样平均近似方法的性能

3.3 混合进化算法

最后分析混合进化算法的性能. 从图 5 可以看出,

随着进化代数的增加, 所获得的资源预留方案逐渐趋向于最优的资源预留方案, 当进化代数$G \geqslant 7$时, 所获得的资源预留方案即为最优的资源预留方案. 因此, 混合进化算法的终止条件选为$G = 20$. 比较本文所提出的混合进化算法和 Hwang 等人所提出的启发式算法的性能, 二者均可获得最优的资源预留方案, 节省了超过 25% 的运营成本, 但是 Hwang 等人的方法需要针对搜索范围内的每个资源预留方案, 确定总的运营成本, 所以求解过程所消耗的时间更长, 并且当最优解不在搜索范围内时不能获得最优的资源预留方案.

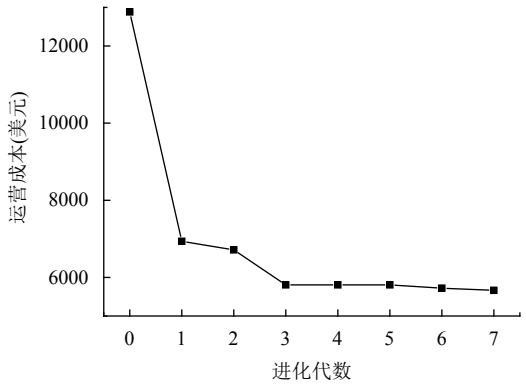

图 5 混合进化算法的进化过程

4 总结

本文针对云资源提供问题, 提出了一种采用随机规划模型的云资源分配算法, 在满足云消费者资源需求的条件下, 最小化其资源使用成本. 首先, 同时考虑按需实例和预留实例, 采用两阶段随机整数规划对云资源提供问题进行建模. 然后, 采用抽样平均近似方法和基于阶段分解的混合进化算法求解云资源提供问题. 基于真实的工作负载数据及 Amazon EC2 的定价模型验证所提出的云资源分配算法. 仿真实验结果表明, 所提出的云资源分配算法能够在较短时间内获得近似最优的云资源预留方案, 有效降低了云消费者的资源使用成本.

参考文献

1 师雪霖, 徐恪. 云虚拟机资源分配的效用最大化模型. 计算机学报, 2013, 36(2): 252–262.
2 赵君, 马中, 刘驰, 等. 一种多目标蚁群优化的虚拟机放置算法. 西安电子科技大学学报 (自然科学版), 2015, 42(3): 173–178, 185.
3 Xiao Z, Song WJ, Chen Q. Dynamic resource allocation using virtual machines for cloud computing environment. IEEE Transactions on Parallel and Distributed Systems, 2013, 24(6): 1107–1117. [doi: 10.1109/TPDS.2012.283]
4 Zhang Q, Zhani MF, Boutaba R, *et al*. Dynamic heterogeneity-aware resource provisioning in the cloud. IEEE Transactions on Cloud Computing, 2014, 2(1): 14–28. [doi: 10.1109/TCC.2014.2306427]
5 Chaisiri S, Lee BS, Niyato D. Optimization of resource provisioning cost in cloud computing. IEEE Transactions on Services Computing, 2012, 5(2): 164–177. [doi: 10.1109/TSC.2011.7]
6 Ran YY, Yang J, Zhang SB, *et al*. Dynamic IaaS computing resource provisioning strategy with QoS constraint. IEEE Transactions on Services Computing, 2017, 10(2): 190–202. [doi: 10.1109/TSC.2015.2464212]
7 Hwang RH, Lee CN, Chen YR, *et al*. Cost optimization of elasticity cloud resource subscription policy. IEEE Transactions on Services Computing, 2014, 7(4): 561–574. [doi: 10.1109/TSC.2013.35]
8 陈俊杰, 周晖, 王伟. 一种低成本的云资源提供算法. 西安交通大学学报, 2017, 51(10): 135–141.
9 Shapiro A, Dentcheva D, Ruszczynski A. Lectures on stochastic programming: Modeling and theory. Philadelphia: Society for Industrial and Applied Mathematics, 2009.
10 Shopova EG, Vaklieva-Bancheva NG. BASIC–A genetic algorithm for engineering problems solution. Computers & Chemical Engineering, 2006, 30(8): 1293–1309.
11 Blickle T, Thiele L. A comparison of selection schemes used in evolutionary algorithms. Evolutionary Computation, 1996, 4(4): 361–394. [doi: 10.1162/evco.1996.4.4.361]
12 Picek S, Jakobovic D. From fitness landscape to crossover operator choice. Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation. Vancouver, Canada. 2014. 815–822.