对于微服务的创建，本算法采用强依赖的概念来表明某个新创建的微服务必须立即连接到已存在的微服务实例上。采用弱依赖的概念来表明某个新创建的微服务能绑定与其关联的微服务或者已经存在的微服务能解绑与其关联的微服务，如下图所示。下图展示了三个微服务：微服务1、微服务2和微服务3。微服务1必须连接到微服务2中，属于强依赖关系。微服务2与微服务3的连接属于弱依赖。当创建了新的微服务1，则必须连接到微服务2提供的接口上。而当创建了新的微服务2则可以连接到微服务3上，也可以不连接到微服务3上。由上述表述可知，弱依赖的概念适用于横向弹性伸缩，通过绑定和解绑相应的微服务来实现横向弹性伸缩。



微服务类型是由五元组构成的集合。其中五元组中的元素含义如下所示：

* 表示可提供的接口，定义为一个从接口到的函数，即。由3.1节可知，代表了可供连接的最大的微服务实例个数。
* 表示强依赖接口，定义为一个从接口到的函数，即。表示了必须连接的微服务实例个数。
* 表示弱依赖接口，与强依赖接口不同的是，弱依赖接口可以不存在，故定义为从一个从接口到的函数，即。
* 表示冲突的接口，其中。

表示资源的消耗量，定义为一个从资源集合到的函数，即。

节点组成了集合，对于每一个在集合中的节点有以下的性质：

* 存在一个表示节点可用资源的函数。本算法使用.res来表示该函数。
* 存在一个数值来表示节点消耗。本算法使用.cost来表示该数值。

配置进行数学定义。一个配置是一个四元组，其中四元组的元素含义如下：

* 表示目前已经部署了的微服务实例集合，显而易见，。
* 表示微服务类型，定义为从已经部署了的微服务实例集合到微服务类型的函数，即。
* 定义为从已经部署了的微服务实例集合到节点的函数，即。
* 定义为一系列绑定的集合，即。

显然，也定义为一个三元组，其中第一个表示需要接口的微服务，第二个表示提供接口的微服务。显然对于，有。本文采用dom表示函数的值域，由于定义为和之间的接口且是需要接口的微服务，而是提供接口的微服务，故有。显然，一个配置就定义了一个微服务系统的状态。

定义了配置后，在本算法中，认为配置存在两种状态：暂时正确和正确。

对于暂时正确的配置拥有以下的性质，如果对任意的节点，有

 ，

显然，上式的含义为，对于任意配置，其节点的可用资源至少要与在节点内部部署的微服务消耗的资源相等。同时对于每一个微服务需要满足以下条件：

* 则意味着对每一个不包含的微服务有且。

一个正确的配置需要满足以下的性质，首先这个配置必须是暂时正确的，其次对于每一个微服务需要满足以下条件：

* 意味着对每一个不包含的微服务有且。

意味着对于每一个不是的微服务有。

假设了微服务存在两种状态，故本算法采用动作的概念对状态的转换进行定义：

* 绑定动作定义为。其中，。绑定的含义是通过接口绑定微服务和。其中是提供接口的微服务。
* 解绑动作定义为，是绑定动作的反方向动作。
* 新建动作定义为，其含义是在节点中定义一个类型的微服务并绑定其强依赖接口到一系列微服务集合，这一绑定过程用函数表示。
* 删除动作定义为。该动作将删除微服务以及与该微服务相关的所有的绑定。

在对算法所需的基本概念进行定义和数学建模后，本节将首先定义弹性伸缩计划，然后引入最优弹性伸缩问题。

一个弹性伸缩计划P定义为配置经过一系列动作使得存在正确的配置，对于任意的有。故可以将弹性伸缩计划P表示为：



在定义了弹性伸缩计划后，定义最优弹性伸缩问题如下所示。给定微服务类型集合、节点集合、初始配置和一个微服务类型()作为输入，输出为弹性伸缩计划P，并满足以下的性质：

* 对于所有的，对任意的有和。
* 对于，存在有。

如果存在这样的一个弹性伸缩计划，在所有可行的计划里面值

最小的弹性伸缩计划P即为最优弹性伸缩问题的解。