算法运行结果对比

在实验中，本研究采用了**GA、禁忌搜索、基于贪心的优化方法、随机算法**与我们提出的 **GSAFO** 算法进行对比：

- **遗传算法（GA）**：遗传算法是一种常用于解决复杂优化问题的方法。在本实验中，我们将路由策略和部署方案视为个体。分别采用单点交叉与随机重新分配路由的方式进行交叉与变异。通过交叉和变异操作产生的不满足约束条件的个体在放入种群之前，将会根据约束条件进行修复。为了确保遗传算法的性能和执行时间，本实验将种群大小配置为 100，进行 50 次迭代，交叉与变异概率分别设置为 0.8 和 0.2。
- **禁忌搜索（Tabu Search）**：该算法从一个初始用户与服务器的路由分配开始，通过**邻域搜索机制**在解空间中进行**迭代**搜索，同时为了**防止**算法重新访问**已经搜索过的解**从而陷入循环，使用禁忌表记录最近访问过的解或解的某些特征。最终通过多次迭代得到了较优解。
- **基于贪心的优化方法（Greedy）**：该算法使用贪心策略将按照**用户优先级**将用户路由到能够**最大化Jain指数**的最优服务器上，以得到最优的路由与部署方案。
- **随机算法（Random）**：该算法将用户请求**随机**路由到云或边缘服务器并进行服务实例的部署。这种方法为本提供了一个随机对照。

a. 固定边缘节点数量(10)，改变用户数量(120/140/160/180)

1. Jain 公平性指对比

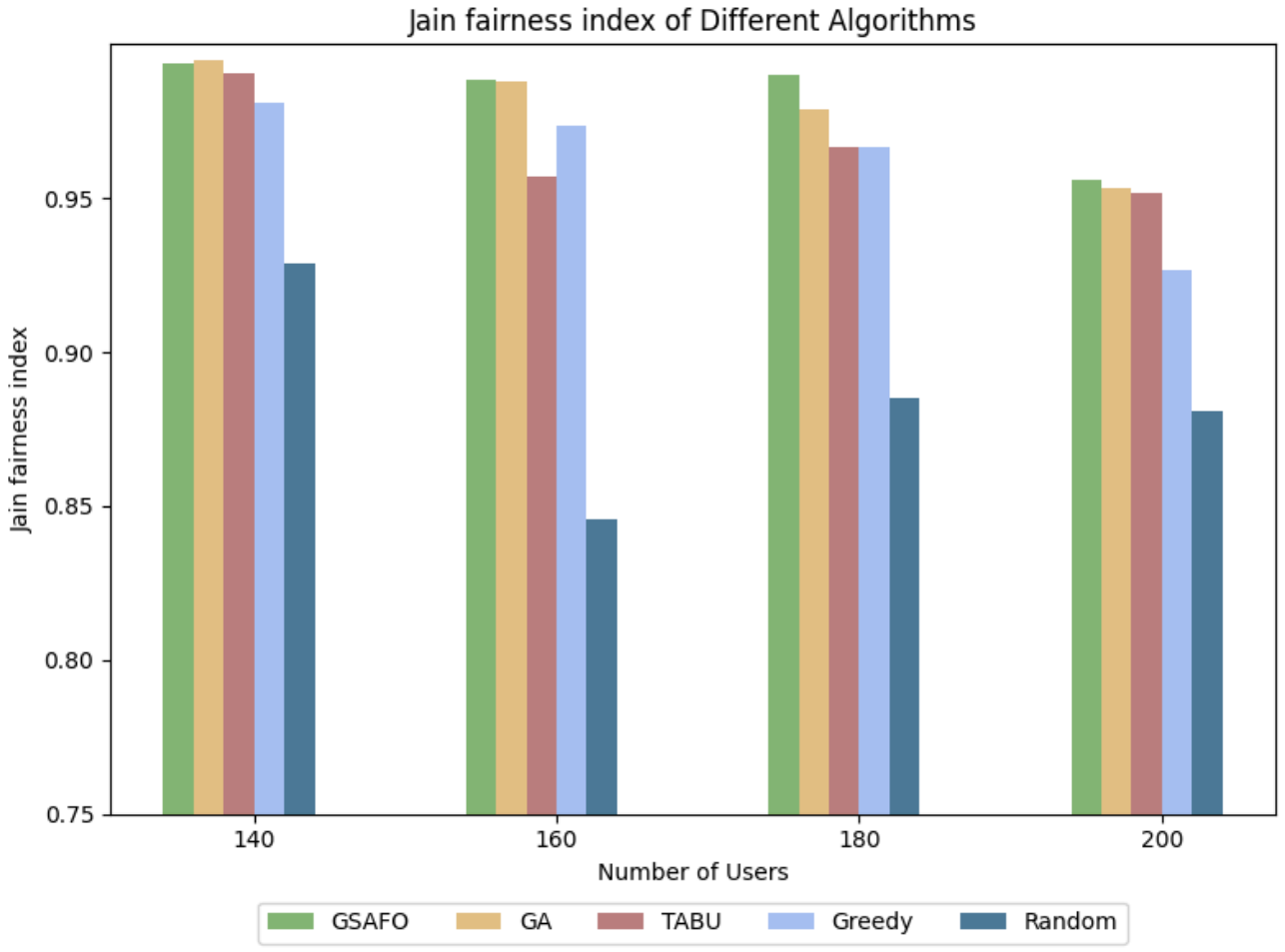

图 1.1

2. 总成本对比

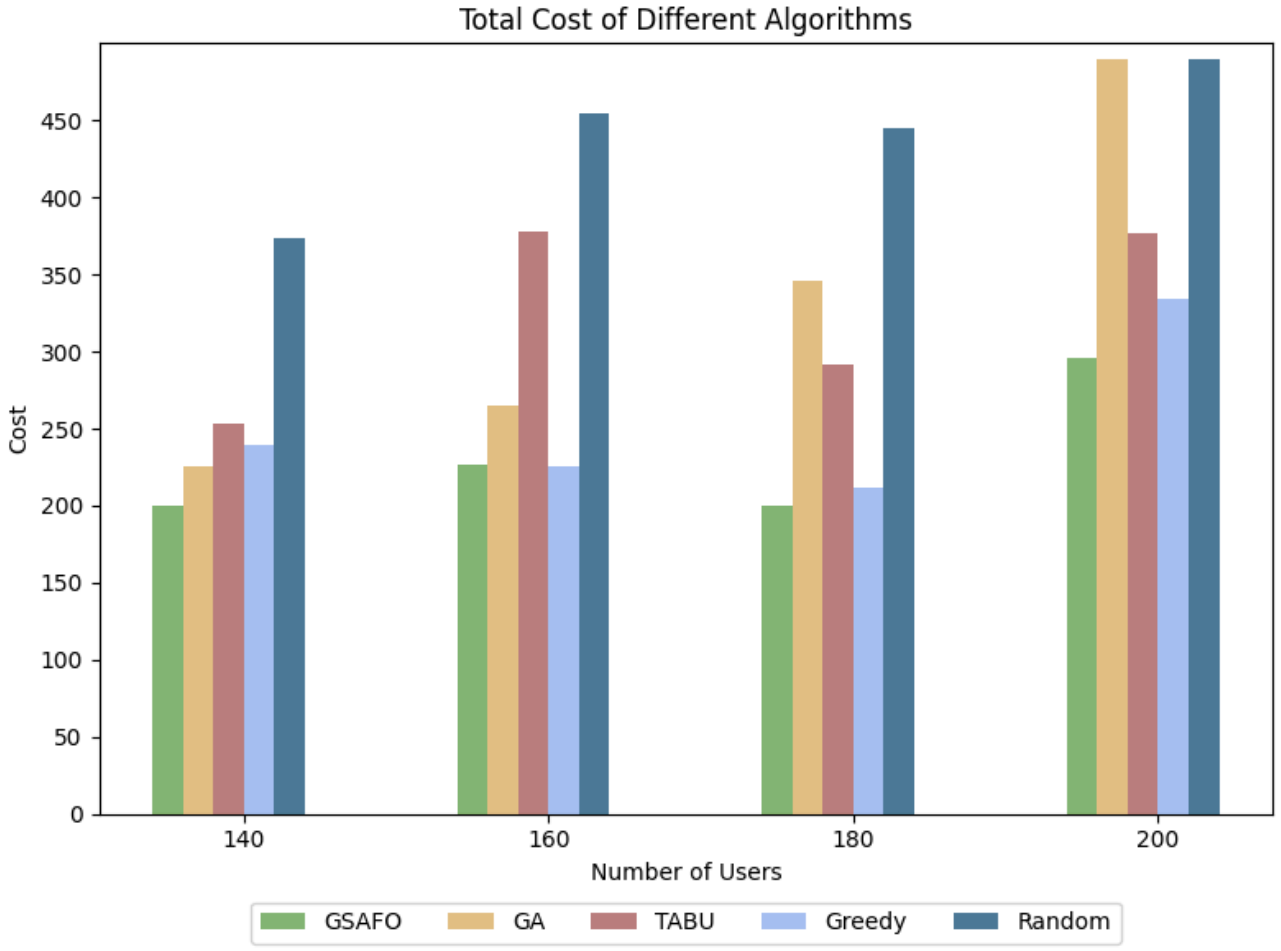

图 1.2

3. 算法运行时间对比

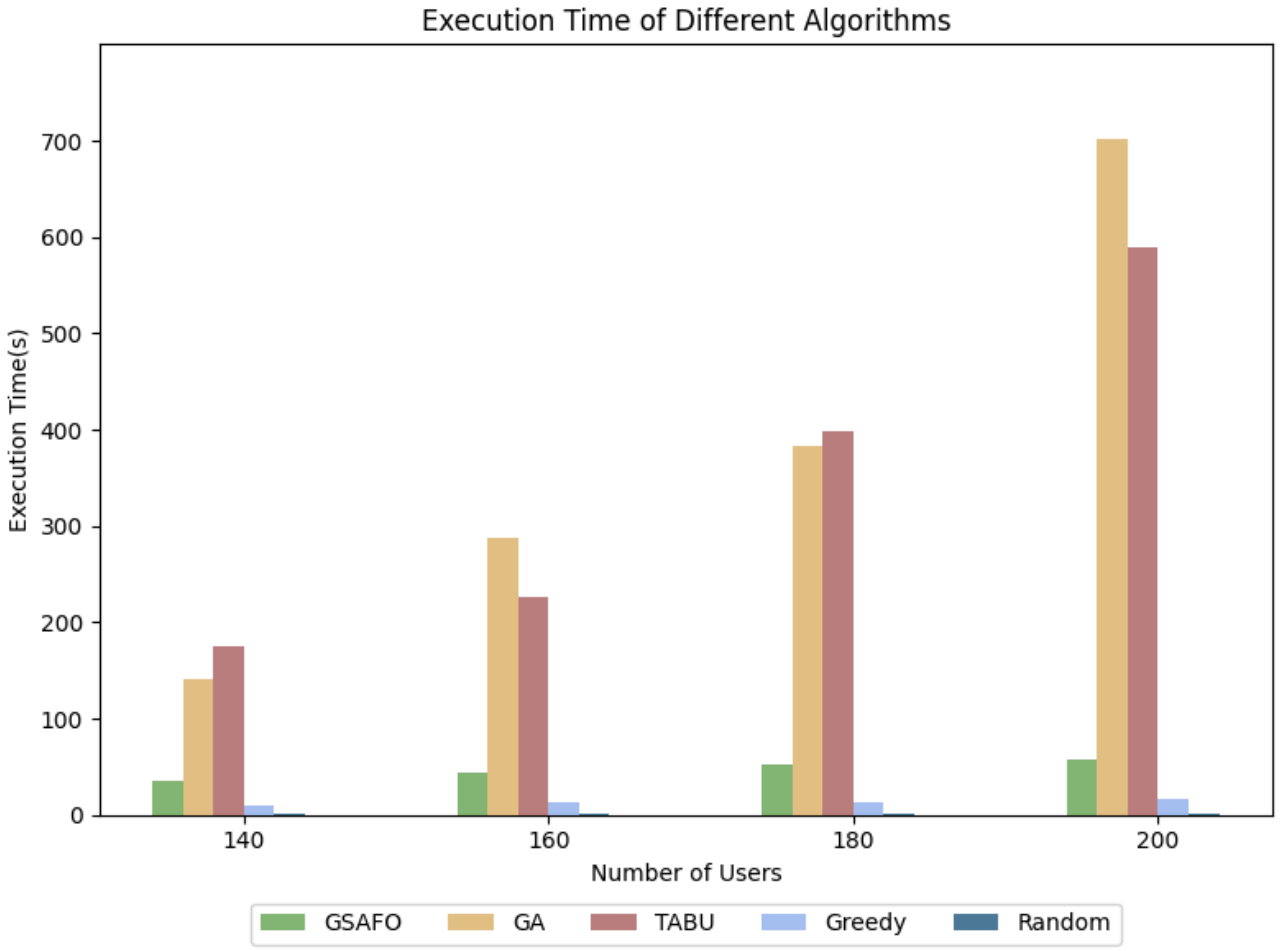

图 1.3

b. 固定用户数量(150)，改变边缘节点数(7/8/9/10)

1. Jain 公平性指数

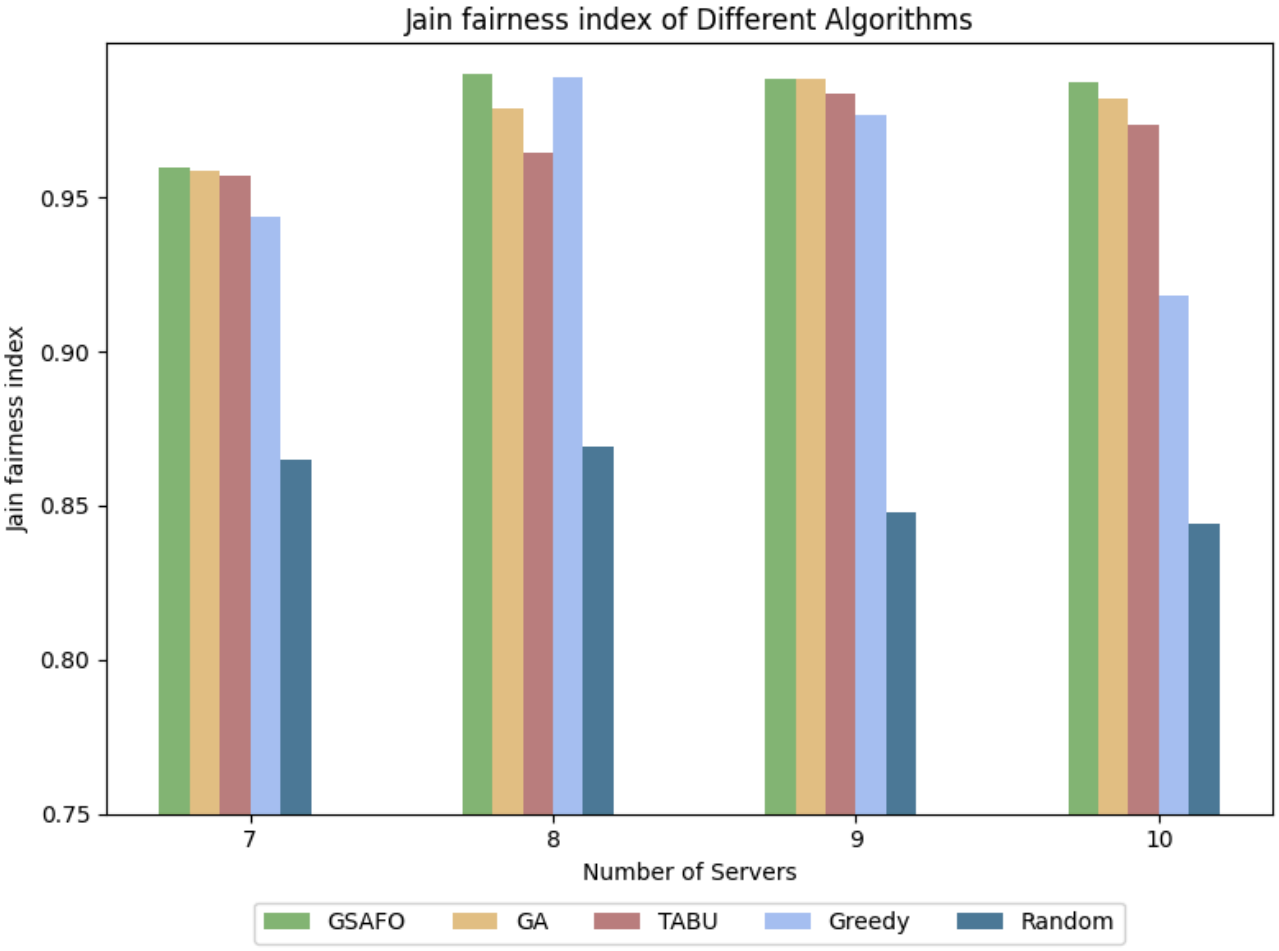
Jain fairness index of Different Algorithms
Jain fairness index
0.95
0.90
0.85
0.80
0.75
7
8
9
10
Number of Servers
GSAFO
GA
TABU
Greedy
Random

图 2.1

2. 总成本

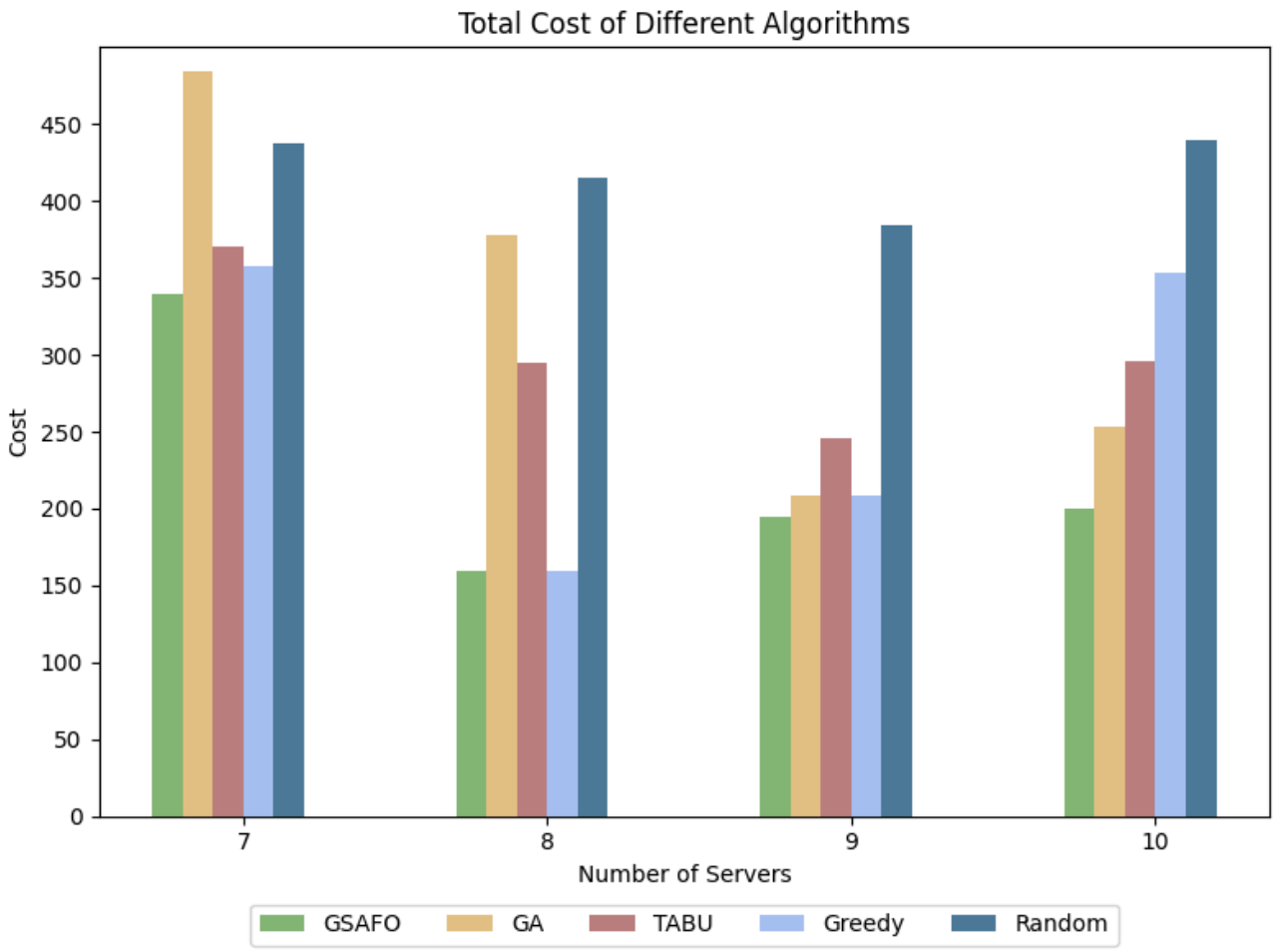
Total Cost of Different Algorithms
Cost
0
50
100
150
200
250
300
350
400
450
7
8
9
10
Number of Servers
GSAFO
GA
TABU
Greedy
Random

图 2.2

3. 算法运行时间

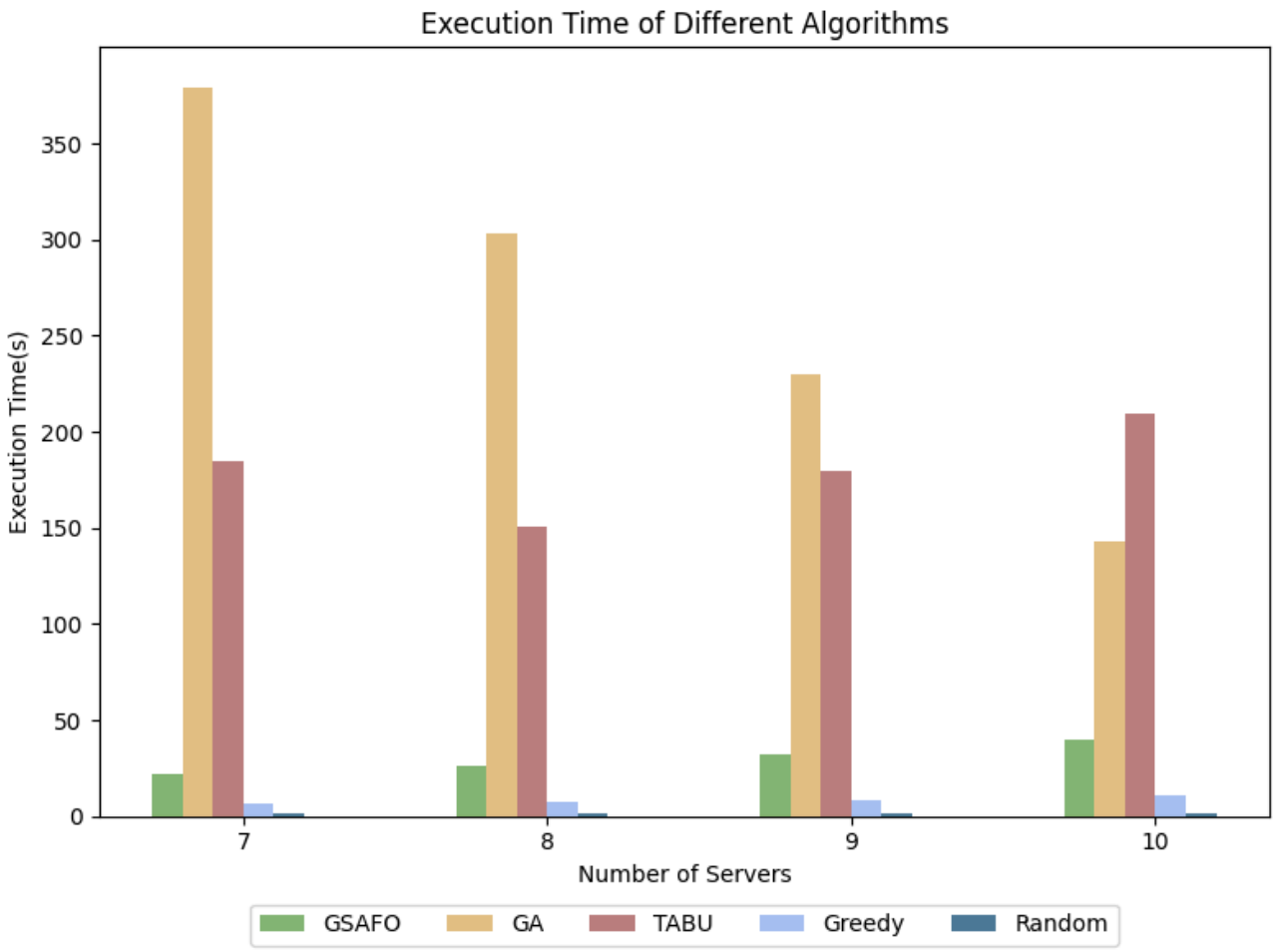

图 2.3

实验结果分析

1. **Jain公平性指数对比：**
   - 如图1.1和2.1所示，GSAFO 算法在不同的实验设置下（即改变**用户数量**或**边缘节点数量**）都能够提供**较高的Jain公平性指数。**Jain指数衡量的是系统的公平性，值越高表示系统的公平性越好。因此，GSAFO算法在提高公平性方面表现优异。
   - 尽管在个别情况下，遗传算法的Jain指数略高，但总体上，GSAFO 算法仍然能提供较为理想的公平性，特别是在**执行时间**方面具有显著优势。
2. **总成本对比：**
   - 图1.2和2.2展示了不同算法在总成本方面的对比。GSAFO 算法能够在保证**较高公平性**的同时，**降低总成本**，这意味着它能更**高效地利用资源。**
   - GSAFO 算法**显著优于** GA 、禁忌搜索算法、基于贪心的优化方法以及随机算法，表现出更强的资源利用效率。
3. **算法运行时间对比：**
   - 从图1.3和2.3可以看出，GSAFO 算法的运行时间远低于遗传算法，这表明 GSAFO 在计算效率上具有明显优势。尤其是在**大规模用户数量**和**较少边缘节点**的情况下（即**系统资源相对紧缺**的情况），GSAFO 算法的优势更加突出。
   - 与基于贪心的优化方法以及随机算法相比，GSAFO 算法的运行时间虽相对较高。然而，GSAFO 算法在系统公平性等关键指标的优化效果上，远远超过了这几种算法。

总结：

GSAFO算法在提升系统公平性（通过Jain指数）、降低总成本和优化算法运行时间方面都表现出了显著优势，尤其在面对**较大规模的用户**和**较少边缘节点数量**时，相比于其他算法，GSAFO 更加高效。