IMMC 中华区建模大赛论文（E 题）

摘要

随着信息技术的飞速发展，数据中心作为现代信息社会的核心基础设施，承担着海量数据存储、处理和分析的重任。然而，其高算力需求也带来了巨大的电力消耗。因此，如何将绿色能源电力与传统电力系统有效结合，实现电力供应的稳定性和可靠性，同时如何合理分配调度算力-电力资源，确保计算任务按时完成并最大化利用绿色能源，降低碳排放和电力成本，是一个值得研究的问题。本研究旨在通过数学建模，优化数据中心的电力和算力调度。

对于问题 1，我们将算力与电力协同调度任务转换为一个以 24 小时总电力成本最低为目标函数的优化模型，通过调整每个时间段 K 内每个小时进行的不同等级的任务数量实现目标函数的优化。通过设置约束条件，使用蒙特卡洛模拟求解，**得到最优算力-电力协调方案中 24 小时的绿色能源总供电 36910 kWh，传统电力总供电 18840 kWh，绿色能源供电占比 66.2%，总电力成本为 33913 元。**

对于问题 2，我们首先进行目标函数的调整改进，为了提高绿色能源供电比例，在原始用电成本最小的目标函数的基础上，添加传统电力的用电量占总用电量的比例最小。**其次，我们进行约束条件的改进。**对于不同时间段 K 内的高紧急任务，仅在该时间段 K 内进行任务数量的分配，确保高紧急任务的实时性不会受到影响。而对于中、低紧急任务，在 24 小时的任意小时段内进行任务的分配，从而大大提高了算力任务协调分配的灵活性。使用蒙特卡洛模拟求解，**得到最优算力-电力协调方案中 24 小时的绿色能源总供电 37220 kWh，传统电力总供电 18530 kWh，绿色能源供电占比 66.8%，总电力成本为 35739 元。**可以看出方案中绿色能源的供电占比明显提高，满足了问题 2 的要求。

对于问题 3，为了实现快速响应并调整算力负载，避免出现电力过剩或不足的情况，即保证每个小时的用电需求量尽量平稳。因此，**我们进行目标函数的改进，在原始用电成本最小的目标函数的基础上，还要使得 24 小时用电量的标准差最小。**对于约束条件，我们采用与问题 2 中同样的约束条件。使用蒙特卡洛模拟求解，**得到最优算力-电力协调方案中 24 小时的绿色能源总供电 35530 kWh，传统电力总供电 20220 kWh，绿色能源供电占比 63.7%，总电力成本为 33473 元，24 小时总用电量的标准差为 1363.9。**可以看出方案满足了问题 3 对于用电量平稳性的要求。

关键词：算力-电力协调优化，目标函数，约束条件，蒙特卡洛模拟；

目录

1 绪论

1.1 背景

随着信息技术的飞速发展，云计算、大数据、人工智能等技术在全球范围内得到了广泛应用，极大地推动了社会的数字化转型。数据中心作为现代信息社会的核心基础设施，承担着海量数据存储、处理和分析的重任，其重要性日益凸显[1]。然而，数据中心的高算力需求也带来了巨大的电力消耗，导致能源消耗上升和碳排放增加。据相关统计，全球数据中心的能耗占全球总能耗的比例逐年上升，这不仅对能源供应提出了巨大挑战，也对环境保护带来了严峻压力[2]。

在全球能源转型的背景下，可再生能源如太阳能、风能等逐渐成为解决能源问题的重要途径。这些绿色能源具有清洁、可再生的特点，能够有效减少碳排放，缓解传统能源的压力[3]。然而，绿色能源的供应存在间歇性和不稳定性，如何将其与传统电力系统有效结合，实现电力供应的稳定性和可靠性，是当前亟待解决的问题。对于数据中心而言，如何在电力供应不稳定的情况下，合理分配和调度算力资源，确保计算任务的按时完成，同时最大化利用绿色能源，降低碳排放和电力成本，成为了一个关键的课题[4]。

本研究旨在通过数学建模，优化数据中心的电力和算力调度，以实现数据中心的高效运行和可持续发展。通过设计合理的电力与算力协同调度模型，能够优化数据中心的能源使用结构，提高绿色能源的使用比例，减少对传统电力的依赖。这不仅有助于降低数据中心的运营成本，还能提高能源利用效率，实现经济效益与环境效益的双赢。研究将考虑不同计算任务的优先级，确保高优先级任务（如紧急计算任务、实时任务）在电力供应不稳定的情况下仍能按时完成，从而提升数据中心的服务质量和服务可靠性。这对于保障关键业务的连续性和稳定性至关重要，尤其是在金融、医疗、交通等对实时性要求较高的领域[5]。本研究将探索如何有效整合绿色能源与传统电力系统，为绿色能源在数据中心的应用提供理论支持和实践指导。通过优化调度策略，能够更好地应对绿色能源的间歇性和不稳定性，促进绿色能源的广泛应用，推动数据中心行业的绿色转型[6]。通过建立数学模型和优化算法，研究将为数据中心的电力和算力调度提供科学的决策依据。这将有助于数据中心管理者更好地应对复杂的调度问题，提高决策的科学性和准确性，提升数据中心的整体运营管理水平[7]。本研究涉及能源管理、任务调度、优化算法等多个领域，其研究成果将为相关领域的研究提供新的思路和方法。例如，优化算法的研究可以为其他资源调度问题提供借鉴，能源管理的研究可以为其他高能耗行业的节能减排提供参考。

总之，本研究不仅具有重要的理论价值，还具有显著的现实意义。通过优化数据中心的电力和算力调度，能够为数据中心行业的可持续发展提供有力支持，同时也为全球能源转型和环境保护做出贡献。

1.2 问题分析

问题 1：设计一个一天（24h）算力与电力协同调度的优化模型，确保数据中心能够根据实时的计算任务需求调整算力资源的分配，并在电力供应条件变化的情况下，动态地调整电力消耗，确保计算任务能够按时完成，避免因电力不足而影响服务质量。

分析：题目中所给数据将每天分为 7 个时间段，每个时间段内包含不同数量的高、中、低紧急度的任务，并且不同任务对应着不同的电力消耗。同时，每个时间段内的绿色能源电力供应量不同，且不同时间段内绿色电力和传统电力的价格也有波动。因此我们将一天（24h）算力与电力协同调度模型转换为一个以电力价格最低为目标函数的优化模型，并设置相应的约束条件，通过调整每个时间段内每个小时进行的不同等级的任务数量实现目标函数的优化。

问题 2：通过电力调度优化，合理调节使用绿色电力和传统电力的比例，降低数据中心的电力成本。同时，团队还需要尽可能提高绿色能源的使用比例，从而达到减排和节省运营成本的双重目标。

分析：我们首先进行目标函数的调整改进，为了提高绿色能源供电比例，在原始用电成本最小的目标函数的基础上，添加传统电力的用电量占总用电量的比例最小。其次，我们进行约束条件的改进。对于不同时间段 K 内的高紧急任务，仅在该时间段 K 内进行任务数量的分配，确保高紧急任务的实时性不会受到影响。而对于中、低紧急任务，在 24 小时的任意小时段内进行任务的分配，从而大大提高了算力任务协调分配的灵活性。最终使得电力成本最低并且绿色能源的使用比例更高。

问题 3：在算力需求和电力供应平衡以及成本优化的基础上设计一个高效的调度算法，在电力供应变化较大的情况下，如何快速响应并调整算力负载，避免出现电力过剩或不足的情况。

分析：为了实现快速响应并调整算力负载，避免出现电力过剩或不足的情况，即保证每个小时的用电需求量尽量平稳。因此，我们进行目标函数的改进，在原始用电成本最小的目标函数的基础上，还要使得 24 小时用电量的标准差最小。对于约束条件，我们采用与问题 2 中同样的约束条件。最终使用蒙特卡洛模拟求解得到满足问题 3 要求的算力-电力协调方案。

1.3 总体思路

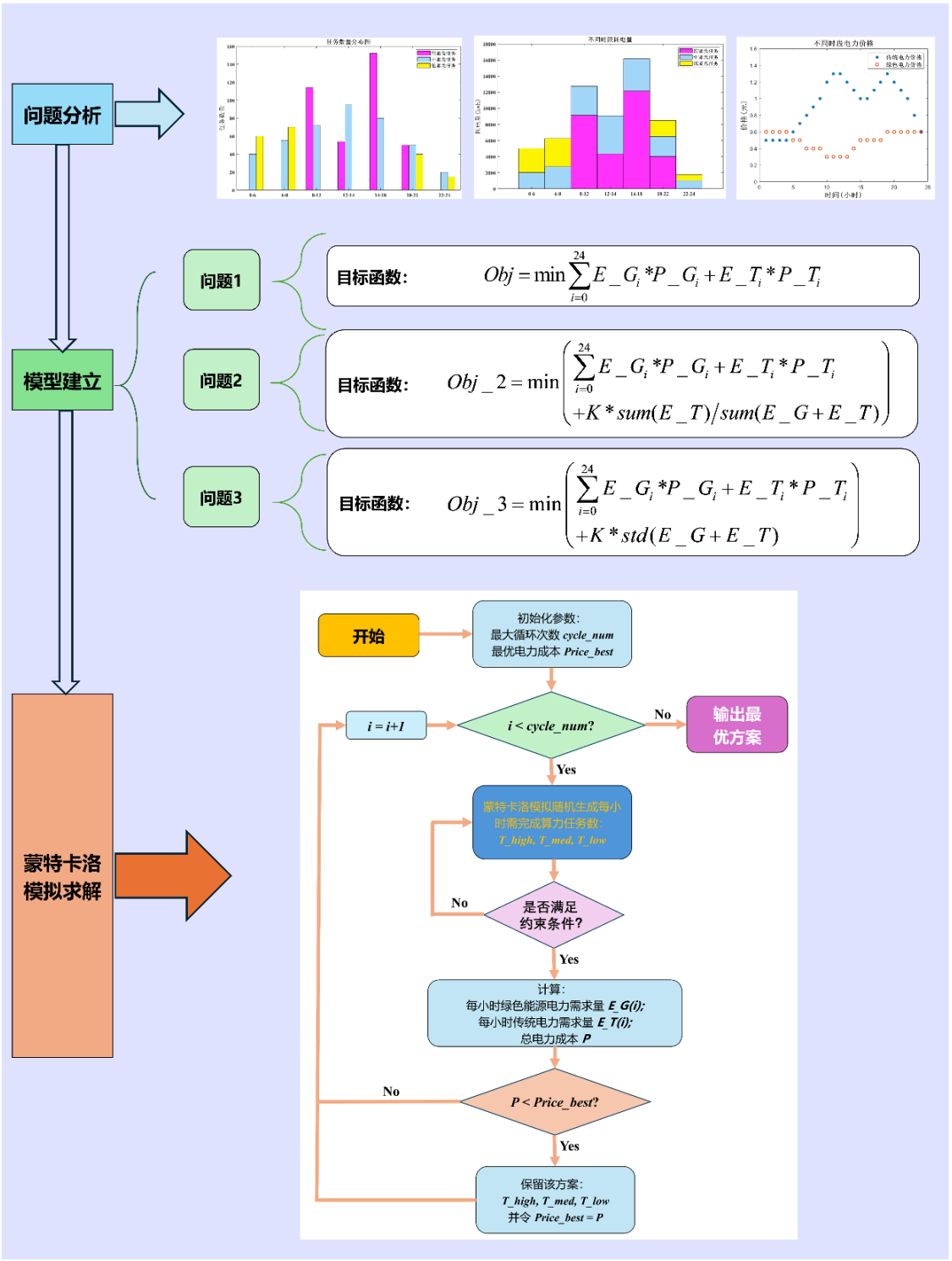

图 1 本题求解过程

1.4 符号说明

本文在模型搭建中涉及的符号及解释说明如下表所示。

表 1 符号说明

符号	释义	单位
K_i	题目数据中一天 24 小时被分成的不同时间段	小时
P_T_i	传统电力在 24 小时不同时间段的价格	元
P_G_i	绿色能源电力在 24 小时不同时间段的价格	元
E_T_i	传统电力在 24 小时不同时间段的需求量	kWh
E_G_i	绿色能源电力在 24 小时不同时间段的需求量	kWh

2 模型假设

为了将实际问题由繁化简，利用数学模型深入研究，本文提出以下模型假设：

假设一：假设传统电力在 24 小时内的供应量是充足的。

解释：通过对题目数据进行分析，发现在夜间时间段内的绿色能源（太阳能、风能等）的发电量较小，因此仅通过绿色能源供电无法完成算力任务的电力需求，因此我们假设传统电力在 24 小时内的供应量是充足的，可以在任何时间段内弥补绿色能源供电量所欠缺的部分。并且我们在任何时间段内优先使用绿色能源供电，仅在绿色能源供电不足时使用传统电力进行补充。

假设二：假设任何等级的算力任务均能在一个小时内完成。

解释：题目中所给数据将每天分为 7 个时间段，每个时间段内包含不同数量的高、中、低紧急度的任务，并且不同紧急程度的任务对应着不同的电力消耗。因此我们将一天（24h）算力与电力协同调度的优化模型转换为一个以电力价格最低为目标函数的优化模型。我们假设任何一个计算任务在一个小时内都可以完成，因此我们可以通过调整每个小时时间段内进行的不同等级的任务数量实现目标函数的优化。

假设三：假设蒙特卡洛模拟在足够多模拟次数后求解的结果近似为最优解。

解释：我们使用蒙特卡洛模拟的算法求解算力与电力协同调度的优化模型，该求解方法会随机生成很多种每个小时时间段内分配不同等级的任务数量的方案，最后求解出总电力成本最低的方案。该求解方法在足够多模拟次数后求解的

结果近似为最优解。

3 问题数据分析

某数据中心电力供应存在不同类型：传统电力和绿色电力。电力供应是有上限的，且电价随着供需波动而变化。电力供应优先考虑绿色能源，但在绿色能源不足时需要使用传统电力。该数据中心有多个计算任务，这些任务的算力需求在不同时间段内变化，并且每个任务的优先级不同。

首先我们将各个数据进行可视化，并进行分析。根据题目中所给数据，可知一天 24 小时被分为了 7 个时间段，分别记为 $K_i, i=1,2,...,7$。下图 1 和图 2 分别可视化了每个时间段内的不同等级的任务数量和需求的电量。

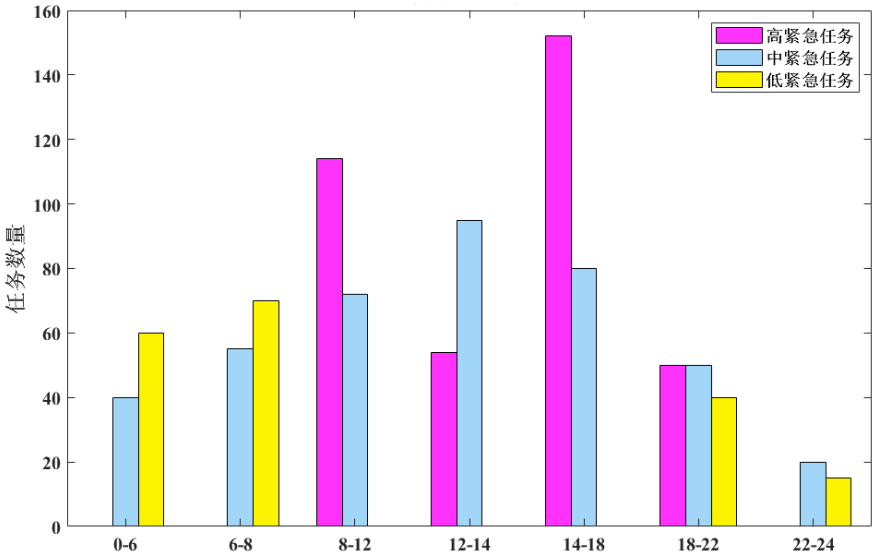

图 2. 24 小时任务数量分布图

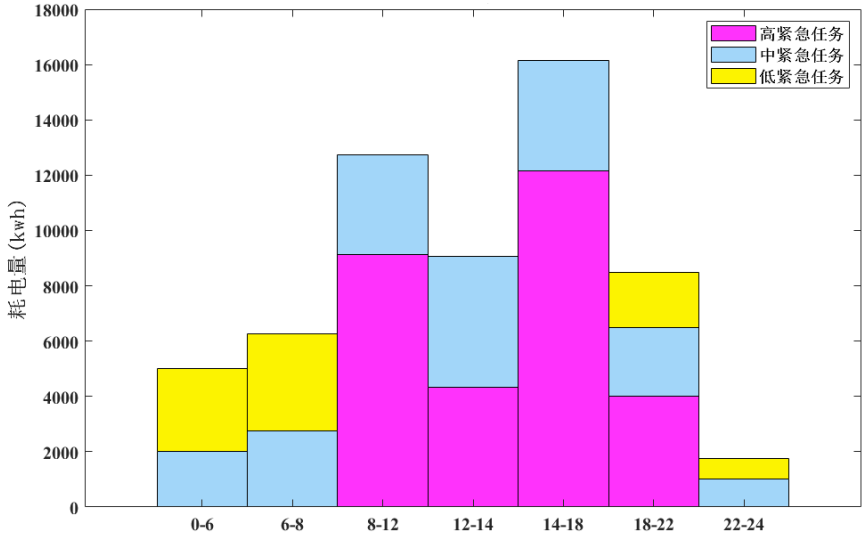

图 3. 24 小时不同时段耗电量

此外，根据题目中数据我们可知传统电力和绿色电力的价格在不同时间段内

是不同的，我们分别记为传统电力价格 $P\_T_i, i=1,2,...,24$ 和绿色电力价格 $P\_G_i, i=1,2,...,24$，并可视化如下图 3 所示。

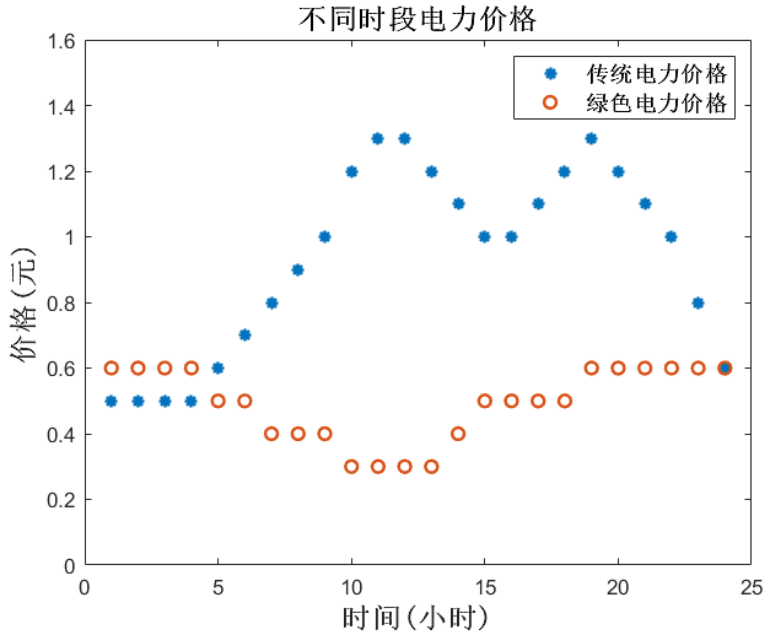

图 4. 24 小时不同时段电力价格

4 算力-电力协同调度优化模型建立与求解

数据中心有多个计算任务，这些任务的算力需求在不同时间段内变化，并且每个任务的优先级不同。任务调度需要考虑不同任务的优先级，确保重要任务（如紧急计算任务）不会受到影响，而不重要的任务则可以根据电力供应情况适当延迟。每个服务器群组的计算能力有限，且每个群组的功耗在运行时是线性的，计算任务的进度与分配的电力和算力直接相关。因此我们需要用数学建模来对某大型数据中心通过合理的电力和算力调度来优化整体运行，达到以下要求

4.1 算力-电力协同调度优化模型的建立

题目中所给数据将每天分为 7 个时间段。每个时间段内包含不同数量的高、中、低紧急度的任务，并且不同紧急程度的任务对应着不同的电力消耗。同时，每个时间段内的绿色能源电力供应量不同，且不同时间段内绿色电力和传统电力的价格也有波动。因此我们将一天（24h）算力与电力协同调度的优化模型转换为一个以电力价格最低为目标函数的优化模型。我们假设任何一个计算任务在一个小时内都可以完成，因此我们可以通过调整每个小时时间段内进行的不同等级的任务数量实现目标函数的优化。

令 $E\_G_i, i=1,2,...,24$ 表示为绿色能源电力在 24 小时不同时间段的需求量，$E\_T_i, i=1,2,...,24$ 表示为传统电力在 24 小时不同时间段的需求量。我们将目标优化函数定义为一天内的总电力成本最低，即绿色能源电力成本和传统电力成本之和，用下式表示：

$$Obj = \min \sum_{i=0}^{24} E\_G_i * P\_G_i + E\_T_i * P\_T_i \tag{1}$$

题目中所给数据将每天分为 7 个时间段,分别记为 $K_i, i=1,2,...,7$ 。。每个时间段内包含不同数量的高、中、低紧急度的任务，并且不同紧急程度的任务对应着不同的电力消耗。因此我们可以通过调整每个小时时间段内进行的不同等级的任务数量实现目标函数的优化。

根据题目中三个问题的不同要求，我们可以设置不同的约束条件，并对目标函数进行不同的改进优化，从而实现电力和算力的最优化调度。

4.2 问题 1 约束条件建立与模型求解

问题 1 要求我们确保数据中心能够根据实时的计算任务需求调整算力资源的分配，并在电力供应条件变化的情况下，动态地调整电力消耗，确保计算任务能够按时完成，避免因电力不足而影响服务质量。

我们假设任何计算任务在一个小时内都可以完成。**因此，我们考虑在不同时间段 K_i 内，对时间段 K_i 内的高、中、低紧急任务在该时间段内进行数量的分配（如对 K_1 时间段的任务进行分配，将会确定 0-1、1-2、2-3、3-4、4-5、5-6 点这 6 个小时段分别分配多少高中低紧急任务）**，从而实现目标函数的最优化。我们假设 $T\_high_i, T\_med_i, T\_low_i$ 分别表示计算得到的分配给给第 i 个小时时间段内高、中、低紧急任务的数量，$N\_high(K_m), N\_med(K_m), N\_low(K_m)$ 分别表示在第 $K_m, m=1,2,...,7$ 个时间段内所需要完成的高、中、低紧急任务的数量。我们根据题目中表 1 所给的数据，可以建立约束条件表示为：

$$\begin{cases}\sum_{i=0}^{6} T\_high_i = N\_high(K_1), \sum_{i=0}^{6} T\_med_i = N\_med(K_1), \sum_{i=0}^{6} T\_low_i = N\_low(K_1) \\ \sum_{i=6}^{8} T\_high_i = N\_high(K_2), \sum_{i=6}^{8} T\_med_i = N\_med(K_2), \sum_{i=6}^{8} T\_low_i = N\_low(K_2) \\ \sum_{i=8}^{12} T\_high_i = N\_high(K_3), \sum_{i=8}^{12} T\_med_i = N\_med(K_3), \sum_{i=8}^{12} T\_low_i = N\_low(K_3) \\ \sum_{i=12}^{14} T\_high_i = N\_high(K_4), \sum_{i=12}^{14} T\_med_i = N\_med(K_4), \sum_{i=12}^{14} T\_low_i = N\_low(K_4) \\ \sum_{i=14}^{18} T\_high_i = N\_high(K_5), \sum_{i=14}^{18} T\_med_i = N\_med(K_5), \sum_{i=14}^{18} T\_low_i = N\_low(K_5) \\ \sum_{i=18}^{22} T\_high_i = N\_high(K_6), \sum_{i=18}^{22} T\_med_i = N\_med(K_6), \sum_{i=18}^{22} T\_low_i = N\_low(K_6) \\ \sum_{i=22}^{24} T\_high_i = N\_high(K_7), \sum_{i=22}^{24} T\_med_i = N\_med(K_7), \sum_{i=22}^{24} T\_low_i = N\_low(K_7) \end{cases}$$

此外，在获得每个小时所分配的高、中、低紧急任务的数量后，由表 1 数据我们可以计算得到每个小时所需要的电量 $E\_hour(i), i = 1,2,...,24$（单位：kWh）：

$$E\_hour(i) = T\_high_i \times 80 + T\_med_i \times 50 + T\_low_i \times 30$$

根据表 2 数据，我们可以得到每个小时的绿色能源电力的最大供应量，因此当该小时的算力需求电量小于绿色能源电力的最大供应量时，我们全部使用绿色能源电力供电。当该小时的算力需求电量大于绿色能源电力的最大供应量，我们将超出的部分使用传统电力供电，约束条件可以表示为：

$$\begin{aligned} &if: E\_hour(i) \le E\_Green(i) \\ &E\_G_i = E\_hour(i) \\ &E\_T_i = 0 \\ &else: \\ &E\_G_i = E\_Green(i) \\ &E\_T_i = E\_hour(i) - E\_Green(i) \end{aligned}$$

其中，$E\_Green(i)$ 指的是第 i 个小时时间段内的绿色能源电力的最大供应量，E_G_i 指的是第 i 个小时时间段内的绿色能源电力的需求量，E_T_i 指的是第 i 个小时时间段内的传统电力的需求量。

我们将上述约束条件带入到式子（1）所示的目标函数中，使用蒙特卡洛模拟算法使得式（1）的值取得最小值，即每天的用电成本最小。

蒙特卡洛模拟求解算法[8]由美国数学家斯坦尼斯拉夫·乌拉姆（Stanislaw

Ulam）和约翰·冯·诺依曼（John von Neumann）提出，又称随机抽样或统计试验方法。该方法采用了利用随机采样解决复杂问题的思想，用计算机实现成千上万次统计模拟或抽样，以获得最优化问题的近似解。

我们使用蒙特卡洛模拟求解该问题的计算流程图如下图 5 所示。

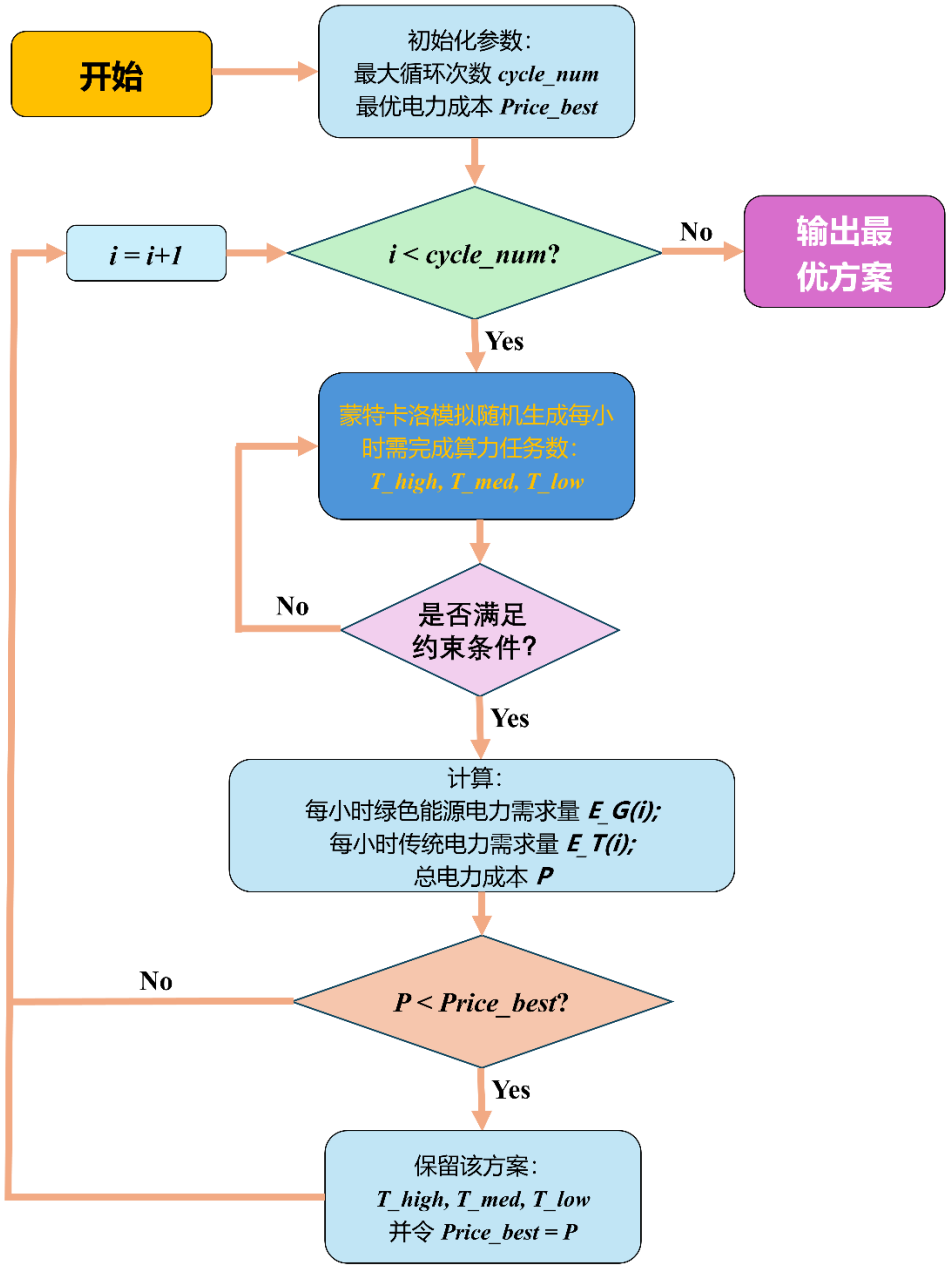

图 5 蒙特卡洛模拟求解流程图

我们设置最大循环次数 ***cycle_num*** 为 1000000 次，设置初始最优电力成本 ***Price_best*** 为 1e10 元，然后使用 MATLAB 编程求解，得到最优算力任务分布方案如下图 6 所示，得到最优电力分布方案如下图 7 所示，详细结果如表 2 所示。

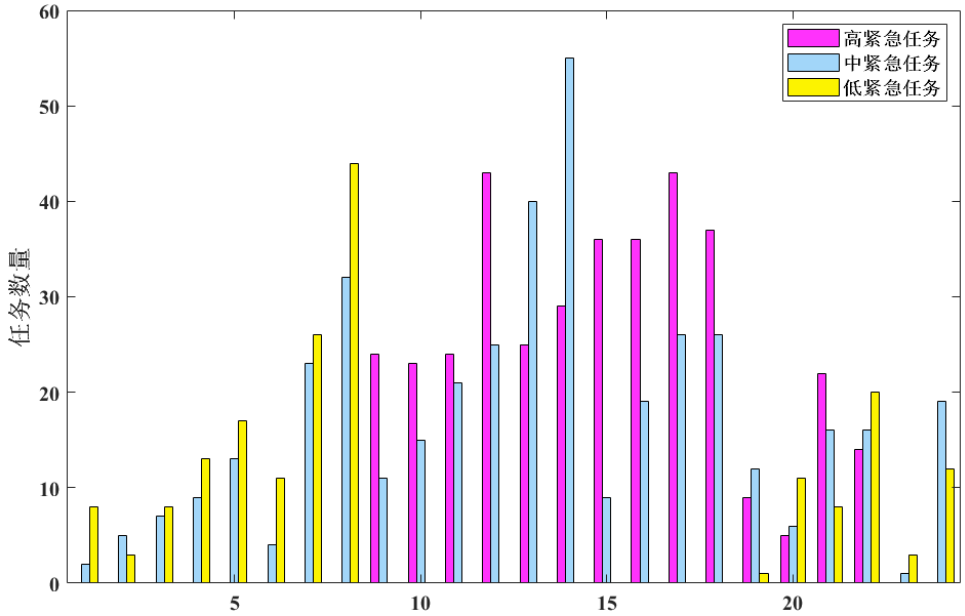

图 6 问题 1 最优算力任务协调分配方案

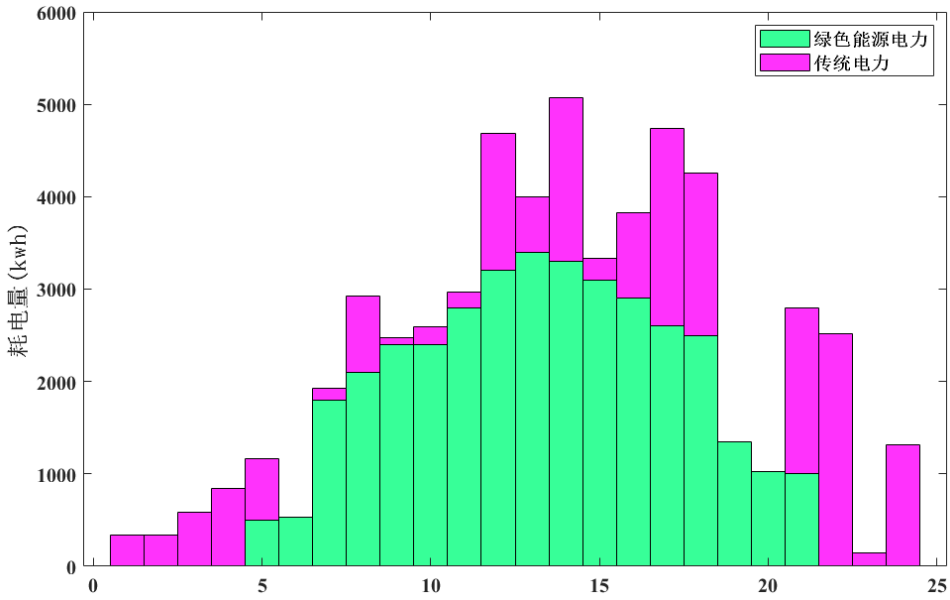

图 7 问题 1 最优电力供应协调分布方案

从图 6 和图 7 的结果中可以看出，算力用电的高峰期集中在每天的 12-18 时间段内，并且在白天的电力分配中绿色能源电力占到供电的大部分，只有在夜间绿色能源无法供电时才会主要使用传统电力供电。表 2 结果中展示了一天 24 小时中每小时时间段内具体的高、中、低紧急任务的分配数量，以及每小时时段内的新能源电力和传统电力的耗电量。**由表 2 可以算出，所得到最优算力-电力协调方案中一天 24 小时的绿色能源总供电 36910 kWh，传统电力总供电 18840 kWh，绿色能源供电占比 66.2%，总电力成本为 33913 元。**

表 2 问题 1 最优算力-电力协调方案

时间段	高紧急方案	中紧急方案	低紧急方案	新能源电力（kWh）	传统电力（kWh）
[0,1]	0	2	8	0	340
[1,2]	0	5	3	0	340
[2,3]	0	7	8	0	590
[3,4]	0	9	13	0	840
[4,5]	0	13	17	500	660
[5,6]	0	4	11	530	0
[6,7]	0	23	26	1800	130
[7,8]	0	32	44	2100	820
[8,9]	24	11	0	2400	70
[9,10]	23	15	0	2400	190
[10,11]	24	21	0	2800	170
[11,12]	43	25	0	3200	1490
[12,13]	25	40	0	3400	600
[13,14]	29	55	0	3300	1770
[14,15]	36	9	0	3100	230
[15,16]	36	19	0	2900	930
[16,17]	43	26	0	2600	2140
[17,18]	37	26	0	2500	1760
[18,19]	9	12	1	1350	0
[19,20]	5	6	11	1030	0
[20,21]	22	16	8	1000	1800
[21,22]	14	16	20	0	2520
[22,23]	0	1	3	0	140
[23,24]	0	19	12	0	1310

4.3 问题 2 约束条件建立与模型求解

问题 2 要求我们通过电力调度优化，合理调节使用绿色电力和传统电力的比例，降低数据中心的电力成本。同时，团队还需要尽可能提高绿色能源的使用比例，从而达到减排和节省运营成本的双重目标。同时，题目中提到任务调度需要考虑不同任务的优先级，确保重要任务（如紧急计算任务、实时任务）不会受到影响，而不重要的任务则可以根据电力供应情况适当延迟。

我们首先针对问题 2 进行目标函数的调整改进，为了提高绿色能源供电的

比例，我们在原始用电成本最小的目标函数的基础上，还要使得传统电力的用电量占 24 小时总用电量的比例最小，更新的目标函数如下式所示：

$$Obj\_2 = \min\left(\begin{array}{l}\sum_{i=0}^{24} E\_G_i * P\_G_i + E\_T_i * P\_T_i \\ +K * sum(E\_T)/sum(E\_G + E\_T)\end{array}\right) \tag{2}$$

其中 K 为比例系数，$sum()$表示求一个序列之和的函数。

其次，我们针对问题 2 进行约束条件的改进。对于不同时间段 K_i 内的高紧急任务，仅在该时间段 K_i 内进行任务数量的分配，确保高紧急任务的实时性不会受到影响。**而对于任意时间段 K_i 内的中、低紧急任务，我们可以在 24 小时的任意小时段内进行任务的分配**，这样大大提高了算力任务协调分配的灵活性，有利于将更多不紧急的算力任务协调到绿色能源供电充足的时间段进行计算，从而尽可能提高绿色能源的使用比例，降低数据中心的电力成本。

我们假设 $T\_high_i, T\_med_i, T\_low_i$ 分别表示计算得到的分配给给第 i 个小时时间段内高、中、低紧急任务的数量，$N\_high(K_m), N\_med(K_m), N\_low(K_m)$ 分别表示在第 $K_m, m = 1,2,...,7$ 个时间段内所需要完成的高、中、低紧急任务的数量。我们根据以上假设，**对于高紧急任务的协调分配可以建立约束条件表示为：**

$$\begin{cases}\sum_{i=0}^{6} T\_high_i = N\_high(K_1), \\ \sum_{i=6}^{8} T\_high_i = N\_high(K_2) \\ \sum_{i=8}^{12} T\_high_i = N\_high(K_3) \\ \sum_{i=12}^{14} T\_high_i = N\_high(K_4) \\ \sum_{i=14}^{18} T\_high_i = N\_high(K_5) \\ \sum_{i=18}^{22} T\_high_i = N\_high(K_6) \\ \sum_{i=22}^{24} T\_high_i = N\_high(K_7)\end{cases}$$

对于中低紧急任务的协调分配可以建立约束条件表示为：

$$\begin{cases}\sum_{i=0}^{24} T\_med_i = \sum_{m=1}^{7} N\_med(K_m) \\ \sum_{i=0}^{24} T\_low_i = \sum_{m=1}^{7} N\_low(K_m)\end{cases}$$

此外，在获得每个小时所分配的高、中、低紧急任务的数量后，由表 1 数据我们可以计算得到每个小时所需要的电量 $E\_hour(i), i = 1,2,...,24$（单位：kWh）：

$$E\_hour(i) = T\_high_i \times 80 + T\_med_i \times 50 + T\_low_i \times 30$$

根据表 2 数据，我们可以得到每个小时的绿色能源电力的最大供应量，因此当该小时的算力需求电量小于绿色能源电力的最大供应量时，我们全部使用绿色能源电力供电。当该小时的算力需求电量大于绿色能源电力的最大供应量，我们将超出的部分使用传统电力供电，约束条件可以表示为：

$$\begin{aligned}&if: E\_hour(i) \le E\_Green(i)\\ &E\_G_i = E\_hour(i)\\ &E\_T_i = 0\\ &else:\\ &E\_G_i = E\_Green(i)\\ &E\_T_i = E\_hour(i) - E\_Green(i)\end{aligned}$$

其中，$E\_Green(i)$ 指的是第 i 个小时时间段内的绿色能源电力的最大供应量，E_G_i 指的是第 i 个小时时间段内的绿色能源电力的需求量，E_T_i 指的是第 i 个小时时间段内的传统电力的需求量。

我们将上述约束条件带入到式子（2）所示的目标函数中，同样使用蒙特卡洛模拟算法求解式（2）的最小值。我们设置最大循环次数 ***cycle_num*** 为 1000000 次，设置初始最优电力成本 ***Price_best*** 为 1e10 元，然后使用 MATLAB 编程求解，得到最优算力任务分布方案如下图 8 所示，得到最优电力分布方案如下图 9 所示，详细结果如表 3 所示。

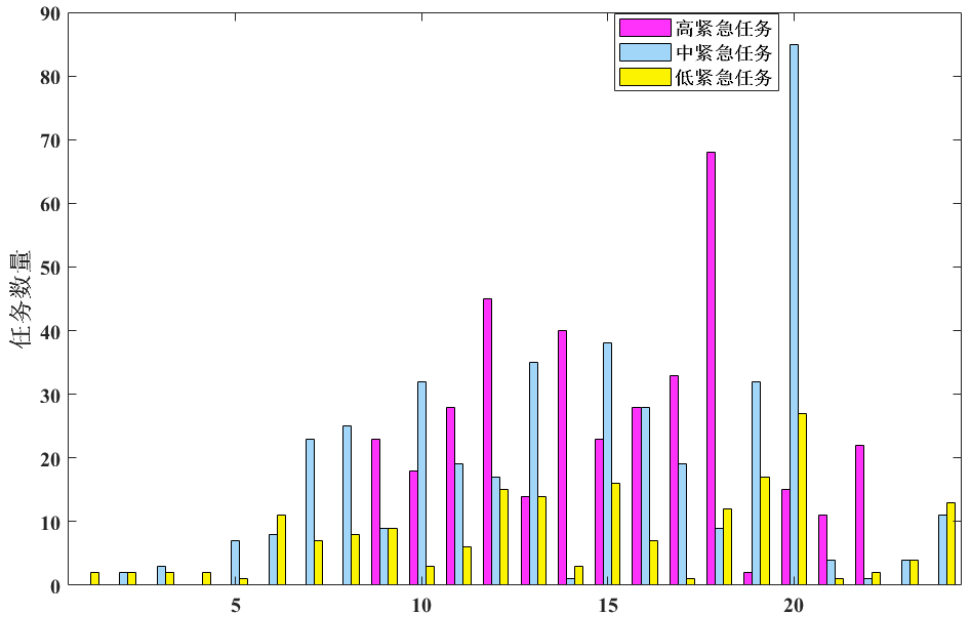

图 8 问题 2 最优算力任务协调分配方案

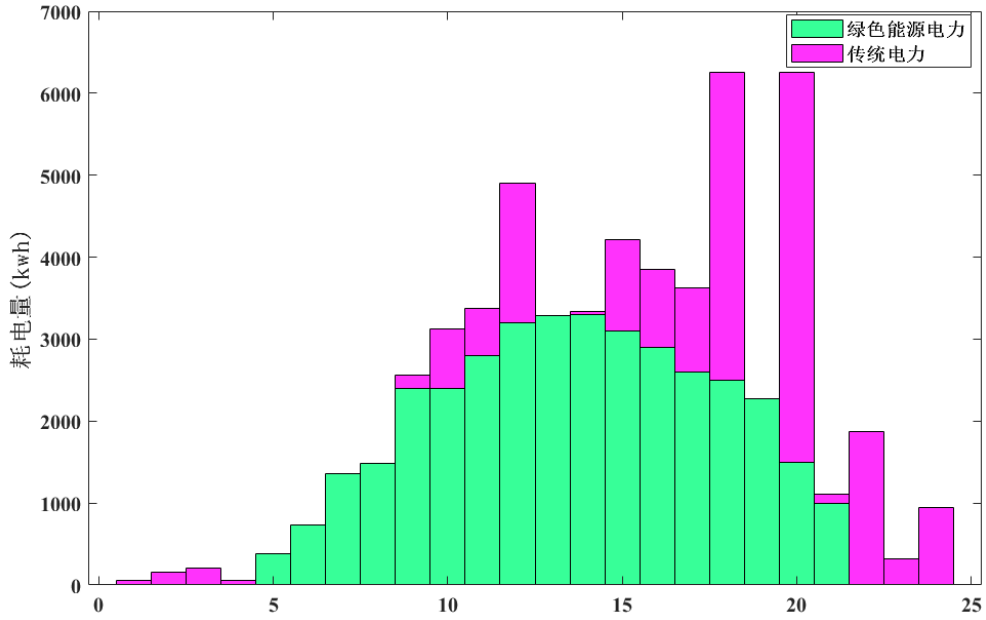

图 9 问题 2 最优电力供应协调分布方案

从图 8 可以看出，最优的算力任务协调分配方案将大部分任务的进行都集中在了白天的时间段，而夜晚时间进行的任务数量不多。这是因为绿色能源电力在白天时间供应充足，而在晚上无法供应，为了提高绿色能源供电的比例，最终得到这样分布的算力任务协调分配方案。从图 9 的结果中可以看出，算力用电的高峰期集中在每天的 10-18 时间段内，并且这个时间段内对于绿色能源电力的使用非常充分，均达到了绿色能源供电的最大值。表 3 结果中展示了一天 24 小时中每小时时间段内具体的高、中、低紧急任务的分配数量，以及每小时时段内的新

能源电力和传统电力的耗电量。**由表 3 可以算出，所得到最优算力-电力协调方案中一天 24 小时的绿色能源总供电 37220 kWh，传统电力总供电 18530 kWh，绿色能源供电占比 66.8%，总电力成本为 35739 元。**

表 3 问题 2 最优算力-电力协调方案

时间段	高紧急方案	中紧急方案	低紧急方案	新能源电力（kWh）	传统电力（kWh）
[0,1]	0	0	2	0	60
[1,2]	0	2	2	0	160
[2,3]	0	3	2	0	210
[3,4]	0	0	2	0	60
[4,5]	0	7	1	380	0
[5,6]	0	8	11	730	0
[6,7]	0	23	7	1360	0
[7,8]	0	25	8	1490	0
[8,9]	23	9	9	2400	160
[9,10]	18	32	3	2400	730
[10,11]	28	19	6	2800	570
[11,12]	45	17	15	3200	1700
[12,13]	14	35	14	3290	0
[13,14]	40	1	3	3300	40
[14,15]	23	38	16	3100	1120
[15,16]	28	28	7	2900	950
[16,17]	33	19	1	2600	1020
[17,18]	68	9	12	2500	3750
[18,19]	2	32	17	2270	0
[19,20]	15	85	27	1500	4760
[20,21]	11	4	1	1000	110
[21,22]	22	1	2	0	1870
[22,23]	0	4	4	0	320
[23,24]	0	11	13	0	940

4.4 问题 3 约束条件建立与模型求解

问题 3 要求我们在算力需求和电力供应平衡以及成本优化的基础上设计一个高效的调度算法，在电力供应变化较大的情况下，如何快速响应并调整算力负载，避免出现电力过剩或不足的情况。

为了实现快速响应并调整算力负载，避免出现电力过剩或不足的情况，也就是需要保证每个小时的用电需求量尽量平稳，即使得每个小时的用电总量是差不多的，该条件可以使用一天 24 小时用电量的标准差来表示。

因此，我们进行目标函数的改进，在原始用电成本最小的目标函数的基础上，还要使得 24 小时用电量的标准差最小，更新的目标函数如下式所示：

$$Obj\_3 = \min\left(\begin{array}{l}\sum_{i=0}^{24} E\_G_i * P\_G_i + E\_T_i * P\_T_i \\ +K * std(E\_G + E\_T)\end{array}\right) \tag{3}$$

其中 K 为比例系数，*std()*表示求一个序列标准差的函数。

对于约束条件，我们采用与问题 2 中同样的约束条件，即**对于不同时间段 K_i 内的高紧急任务，仅在该时间段 K_i 内进行任务数量的分配**，确保高紧急任务的实时性不会受到影响。**而对于任意时间段 K_i 内的中、低紧急任务，我们可以在 24 小时的任意小时段内进行任务的分配。**

我们将上述约束条件带入到式子（3）所示的目标函数中，同样使用蒙特卡洛模拟算法求解式（3）的最小值。我们设置最大循环次数 ***cycle_num*** 为 1000000 次，然后使用 MATLAB 编程求解，得到最优算力任务分布方案如下图 10 所示，得到最优电力分布方案如下图 11 所示，详细结果如表 4 所示。

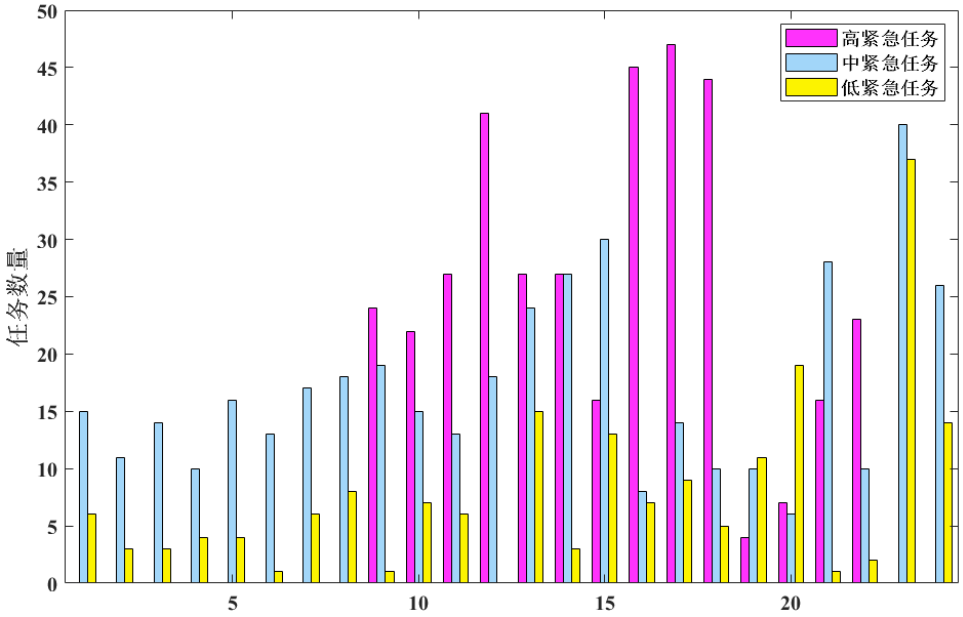

图 10 问题 3 最优算力任务协调分配方案

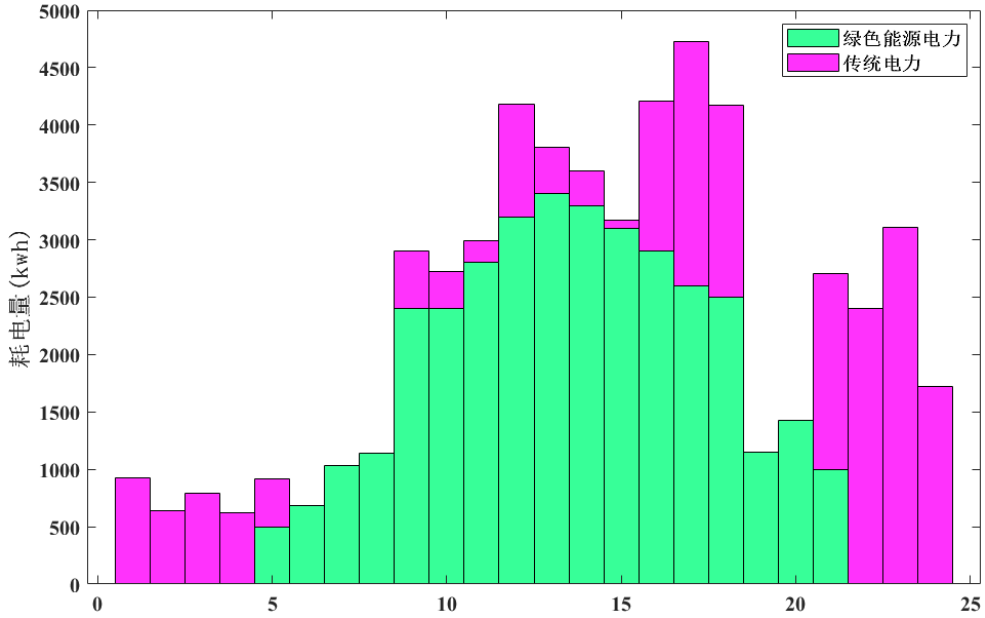

图 11 问题 3 最优电力供应协调分布方案

从图 10 可以看出，最优的算力任务协调分配方案将中低紧急的任务都均匀分布在了一天 24 小时的每个时间段，从而使得在保证电力成本最低的基础上使得每个小时的用电需求量是平稳的。从图 11 的结果中可以看出，在白天时段每个小时的用电需求量是平稳的，相差不多的，而在夜晚时段也保证了电量的使用，避免出现电力过剩或不足的情况。**由表 4 可以算出，所得到最优算力-电力协调方案中一天 24 小时的绿色能源总供电 35530 kWh，传统电力总供电 20220 kWh，绿色能源供电占比 63.7%，总电力成本为 33473 元，24 小时总用电量的标准差为 1363.9。**

表 4 问题 3 最优算力-电力协调方案

时间段	高紧急方案	中紧急方案	低紧急方案	新能源电力（kWh）	传统电力（kWh）
[0,1]	0	15	6	0	930
[1,2]	0	11	3	0	640
[2,3]	0	14	3	0	790
[3,4]	0	10	4	0	620
[4,5]	0	16	4	500	420
[5,6]	0	13	1	680	0
[6,7]	0	17	6	1030	0
[7,8]	0	18	8	1140	0
[8,9]	24	19	1	2400	500
[9,10]	22	15	7	2400	320

[10,11]	27	13	6	2800	190
[11,12]	41	18	0	3200	980
[12,13]	27	24	15	3400	410
[13,14]	27	27	3	3300	300
[14,15]	16	30	13	3100	70
[15,16]	45	8	7	2900	1310
[16,17]	47	14	9	2600	2130
[17,18]	44	10	5	2500	1670
[18,19]	4	10	11	1150	0
[19,20]	7	6	19	1430	0
[20,21]	16	28	1	1000	1710
[21,22]	23	10	2	0	2400
[22,23]	0	40	37	0	3110
[23,24]	0	26	14	0	1720

5 模型优缺点分析

5.1 模型优点分析

1. 本研究将一天（24h）算力与电力协同调度的优化模型转换为一个以 24 小时总电力成本最低为目标函数的优化模型，通过调整每个时间段 K 内每个小时进行的不同等级的任务数量实现目标函数的优化。

2. 本研究针对不同的算力-电力调度需求，进行不同的目标函数改进和约束条件设计。如对于问题 2，我们首先进行目标函数的调整改进，在原始用电成本最小的目标函数的基础上，添加传统电力的用电量占总用电量的比例最小。其次，我们进行约束条件的改进。对于中、低紧急任务，在 24 小时的任意小时段内进行任务的分配，从而大大提高了算力任务协调分配的灵活性。

3. 本研究使用蒙特卡洛模拟求解离散约束的优化问题。该算法适用范围广，灵活性高，可以处理各种复杂的非线性问题，不需要对问题进行简化或者做出任何假设。并且可以处理高维问题，不受维数限制，模拟次数越多，结果越精确。

5.2 模型缺点分析

1. 使用蒙特卡洛模拟求解，其随机性导致方差较大：由于随机性，模拟结果的方差较大，需要进行大量的模拟才能得到可靠的结果。并且模拟的速度较慢，需要大量的计算资源和时间。

6 参考文献

[1] Beloglazov, A., & Buyya, R. (2012). Energy efficient allocation of virtual machines in cloud data centers. In 2012 IEEE/ACM 3rd International Conference on Cyber, Physical and Social Computing (pp. 1-8). IEEE.

[2] Shehabi, A., Smith, S., & Masanet, E. (2016). United States data center energy usage report. Lawrence Berkeley National Laboratory.

[3] Al-Fares, M., Loukissas, A., & Vahdat, A. (2008). A scalable, commodity data center network architecture. In ACM SIGCOMM Computer Communication Review (Vol. 38, No. 4, pp. 63-74). ACM.

[4] Zhang, X., & Shen, X. S. (2013). Energy-efficient resource management in cloud data centers: A survey. IEEE Communications Surveys & Tutorials, 15(4), 1738-1753.

[5] Luo, J., & Li, K. (2015). Energy-efficient task scheduling for cloud data centers with renewable energy sources. IEEE Transactions on Parallel and Distributed Systems, 26(10), 2684-2697.

[6] Li, K., & Luo, J. (2014). Energy-efficient task scheduling in cloud data centers with renewable energy sources. In 2014 IEEE 28th International Parallel & Distributed Processing Symposium (pp. 1-10). IEEE.

[7] IBM. (2023). Data centers: The backbone of the digital economy. IBM Think. https://www.ibm.com/think/topics/data-centers

[8] Landau DP, Binder K. A Guide to Monte Carlo Simulations in Statistical Physics. 2nd ed. Cambridge University Press; 2005.

7 附录代码

（1）生成 n 个和为 m 的随机正整数函数

```
function PH=RandomSum2(N,m)
%随机生成n个和为m的正整数
    PH = zeros(1,N);
    SUM_powini = m;
    % 初始化总负荷
    counter = 0;
    while (PH(N)<=0) && (counter <100)
        sumtemp = floor(SUM_powini/N);
        for j=1:(N-1)
            PH(j) = sumtemp.*abs(rand());
            sumtemp = floor((SUM_powini - PH(j))/(N-j));
        end
        PH(N) = SUM_powini - sum(PH(1:N-1));    % 初始化负荷
        counter = counter + 1;
    end
    total=0;
    for i=1:N-1
      PH(i)=round(PH(i));
      total=total+PH(i);
    end
    PH(N)=m-total;
end
```

（2）模型建立与求解代码

```
%导入电价
price_green=[0.6 0.6 0.6 0.6 0.5 0.5 0.4 0.4 0.4 0.3*ones(1,4) 0.4 0.5*ones(1,4) 0.6*ones(1,6)];
price_tradition=[0.5*ones(1,4) 0.6 0.7 0.8 0.9 1.0 1.2 1.3 1.3 1.2 1.1 1.0 1.0 1.1 1.2 1.3 1.2 1.1 1.0 0.8 0.6];
Green_limit=[0*ones(1,4) 500 1400 1800 2100 2400 2400 2800 3200 3400 3300 3100 2900 2600 2500 2300 1500 1000 0*ones(1,3)];

%任务数量
High_Tasks=[0 0 114 54 152 50 0];
Medium_Tasks=[40 55 72 95 80 50 20];
Low_Tasks=[60 70 0 0 0 40 15];

%%
%蒙特卡洛求解
%第一问  任务只在每个时间段内分配
cycle_num=100000; %蒙特卡洛循环次数
%保留最优解
E_best_price=1e10; %保存最优惠的价格
T_high_best=zeros(1,24);
T_med_best=zeros(1,24);
T_low_best=zeros(1,24);
E_tradition_best = zeros(1,24);
E_new_best = zeros(1,24);

for cycle_i=1:cycle_num
    E_tradition = zeros(1,24); %传统电力 kwh
    E_new = zeros(1,24); %新能源电力
    %0-6点
    T1_high = RandomSum2(6,High_Tasks(1));
    T1_med = RandomSum2(6,Medium_Tasks(1));
    T1_low = RandomSum2(6,Low_Tasks(1));
    %6-8点
    T2_high = RandomSum2(2,High_Tasks(2));
    T2_med = RandomSum2(2,Medium_Tasks(2));
    T2_low = RandomSum2(2,Low_Tasks(2));
    %8-12点
    T3_high = RandomSum2(4,High_Tasks(3));
    T3_med = RandomSum2(4,Medium_Tasks(3));
    T3_low = RandomSum2(4,Low_Tasks(3));
    %12-14点
    T4_high = RandomSum2(2,High_Tasks(4));
    T4_med = RandomSum2(2,Medium_Tasks(4));
    T4_low = RandomSum2(2,Low_Tasks(4));
    %14-18点
    T5_high = RandomSum2(4,High_Tasks(5));
    T5_med = RandomSum2(4,Medium_Tasks(5));
```

```
T_high=[T1_high T2_high T3_high T4_high T5_high T6_high T7_high];
T_med=[T1_med T2_med T3_med T4_med T5_med T6_med T7_med];
T_low=[T1_low T2_low T3_low T4_low T5_low T6_low T7_low];
%E_hour= T_high(hour)*80 + T_med(hour)*50 + T_low(hour)*30;
E_hour= T_high*80 + T_med*50 + T_low*30;
%计算每一小时需要的传统电量和新能源电量
for hour=1:24

    if E_hour(hour) > Green_limit(hour) %该小时内需要的总电量大于新能源发的电量，则超出的部分使用传统发电
        E_new(hour) = Green_limit(hour);
        E_tradition(hour) = E_hour(hour) - E_new(hour);
    else %否则全部使用风力发电
        E_new(hour) = E_hour(hour);
    end
end

Total_price = sum(E_new.*price_green)+sum(E_tradition.*price_tradition); %计算该方案的总电费
```

```
    Total_price = sum(E_new.*price_green)+sum(E_tradition.*price_tradition); %计算该方案的总电费
    if Total_price<E_best_price
        E_best_price = Total_price;
        T_high_best = T_high;
        T_med_best = T_med;
        T_low_best = T_low;
        E_tradition_best = E_tradition;
        E_new_best = E_new;
    end
end
```