郑老师、师兄师姐以及各位同门大家上午好。我这次组会要汇报的主题是基于逻辑的Benders分解算法及其应用。

首先，非常感谢郑老师还有各位师兄师姐以及同门上学期关于选题对我的指导和建议，这些指导和建议对我帮助非常大。在这些指导和建议下，我和郑老师进行了交流，最终将我的选题确定为《软煤质约束下快速周转筒仓码头的多资源联合调度》。关于这个选题，大家可能会有一些疑问，比如说，这个题目中的Soft Coal Quality Constraint也就是软煤质约束它是个什么东西。因此，我有必要介绍一下我的这个选题。从问题界定的角度出发，这个选题实际上可以用一句话概括——就是在煤炭出口码头做航道调度+泊位分配+筒仓组运营的集成优化。那这个东西有什么做的价值呢？首先，我们认为在环境监管政策日益严苛的时代背景下，筒仓堆场很有可能是未来堆场形态的一种趋势，特别对于是高污染的干散货堆场，筒仓封闭式的存储结构可以有效缓解环境污染问题，这是前景。就现状来讲，我国最大的煤炭下水港黄骅港早在2022年就有超过一半的煤炭下水量是通过它3,4期工程那4排12列，共计48个筒仓构成的筒仓码头实现的，并且它们要新建的5期工程也是准备建设筒仓堆场而不是露天堆场。以上是实践层面，在理论研究方面，与集装箱码头相比，干散货码头本身研究的程度就不够深入，并且在堆场这个层面，以往的研究基本都集中在露天堆场环境下。与露天堆场相比，筒仓堆场在结构和运营模式上存在诸多差异，这就带来了理论和实践层面的GAP。我们进一步对稀少的干散货码头筒仓堆场的文献进行分析认为这个领域以往的研究存在很多局限性，有很大的改进空间，这些GAP主要存在于三个方面：（1）这些研究对于库存的动态性刻画不足，多数研究假设筒仓容量无限或货物充足，忽略了筒仓库存补给过程和库存变化对作业可行性的影响；（2）对于DST运营模式下的煤炭出口码头，原煤混配是必要的作业程序，极少有文献考虑这一点，而那些考虑了原煤混配的文章往往又过于简化了原煤混配这个过程，普遍假设配煤方案预先已知且比例固定；（3）筒仓组运营与航道调度等海侧作业的集成研究匮乏，筒仓码头的高效运营需要海侧、岸侧、堆场侧的协调，但现有研究多聚焦于岸侧与堆场侧，对船舶进出港过程的考虑较为简化。在现实中航道调度、泊位分配和筒仓组运营三个决策过程是相互交织、紧密耦合的。站在整体的角度，考虑单方面的决策都可能是次优甚至是不可行的。以上就是确定这个选题的主要原因。最后，我想谈一谈标题当中的这个Soft Coal Quality Constraints是个什么东西，这也是我想把这个选题做的比较有特色的点之一。回顾以往做原煤混配的文献，我们发现它们往往是假定原煤混配方案已知，什么意思？比如说我作为港口，我事先知道你可以接受哪些原煤混配方案，比方说原煤A：原煤B=1:2，或者是原煤A：原煤C=1:3等等。我作为港口在知道这些信息的情况下去协调码头的作业，这种研究方式在假设堆场库存充足的情况下是ok的，没有问题，具体用哪种原煤混配方案可以选择嘛。但在实际当中，堆场相关参与混配的原煤的场存能够做到一直都是充足的吗？这是一个关键问题。如果现实中做不到，那像这样高精度的原煤混配方案的模型优化结果就必然会带来一个问题——这个问题就是漫长的船等货现象。这种漫长的等待对于筒仓码头可能是致命的，由于封闭的存储环境，筒仓内的煤炭非常容易积累热量，发生自燃现象，因此筒仓堆场需要实现煤炭的快速周转。据统计，一块儿原煤从它到黄骅港堆场到离开黄骅港的平均时长不会超过10天，筒仓堆场更快。这个影响是一方面，另一方面，船等货现象也会对其他船舶的进出港以及靠泊作业产生影响，使得优化结果次优甚至实际上不可行。基于这一点，我们首次提出了软配煤策略的概念，这个策略就是说——货主，你不要再告诉我具体的煤炭混配方案了，我作为港口我希望知道你理想状态下的那种合同煤的理化性质，以你及能够接受的理化性质浮动范围和锅炉燃烧要求，这个煤具体怎么配你不要管了，我来帮你设计。我作为港口，我会告诉货主，你放心，只要我堆场的库存是充足的，我一定还

是会配出你理想的那种合同煤的，但如果我库存紧张，那我配的煤可能会和你最理想的那种有偏差，你得理解我，我会给你一些补偿。在这种策略下，我们引入了原煤混配的柔性，可以最大程度上利用堆场现有场存，实现原煤和船舶的快速周转。像国能集团这种上下游一体化的企业，我认为这种策略是有可能实现的。以上就是我选题的大概内容，根据郑老师的建议，是希望把硬配煤策略和软配煤策略的优化结果做一个对比，我目前的进度是将硬配煤策略下的模型搭建完毕了，并且使用求解器做了测试。接下来，我会开始尝试搭建软配煤策略下的模型并设计相应的求解算法。注意到在求解算法方面，干散货领域的研究由于模型比较复杂，基本上是以启发式算法为主的，做精确解的文章非常少，这个领域现在可能也确实是需要更多做精确解的文章的，特别是基于分解的算法，王帅安老师在他25年关于干散货码头运营的综述中也提到了这一点。因此，我想挑战一下，看看能不能用LBBD把这个问题搞出来。最近这段时间看了一些LBBD的书籍和论文，感觉脑袋非常混乱，这次也是借着组会的机会整理整理头绪。ok，下面进入正题。

关于Benders分解和基于逻辑的Benders分解，在我学习的过程中主要是遇到了下面这三组问题，希望这次我能把他们描述清楚，如果我说的不对请随时打断我：（1）Benders分解和行生成是什么关系？Benders 1962年提出的经典Benders分解的思路是什么，经典Benders分解在应用上有什么局限性？（2）Hooker提出的Inference dual是什么东西，他为什么要提出这个东西？LBBD和经典Benders分解有什么区别？LBBD收敛到最优解的充分条件是什么？Inference dual是否是强对偶对LBBD的收敛性有影响吗？（3）第一篇用LBBD做考虑准备时间的并行机调度问题的论文是怎么用LBBD的？近几年干散货码头运营相关论文是怎么用LBBD的，LBBD的设计难点在哪？因为经典Benders分解是基于逻辑的Benders分解的基础，它们处理问题的核心思想都是一样的，这个思想可以描述为——我们首先对原问题做某种程度的松弛，将得到的新问题定义为主问题，对原问题做某种程度的限制，比方说固定一组变量，将得到的新问题定义为子问题。因为主问题是原问题的松弛，所以主问题始终给原问题提供下界（如果原问题是最小化问题的话），然后我们再通过某种手段根据子问题解的情况，把我们松弛掉的那部分信息慢慢拿回来。这里提到的手段，实际上就是往主问题里面加割，也就是加约束。以上是相通的部分，而经典benders分解和逻辑benders分解的差异实际上也还是在这个加割的手段上，经典Benders分解割的理论依据是线性规划对偶，而逻辑benders分解割的理论依据是Inference dual，也就是推理对偶。时间关系，经典benders分解和逻辑benders分解相通的部分我们就略过了，如果去写一个程序，那么经典benders分解和基于逻辑的benders分解最外层的大框架是完全一样的，区别就在割的设计上面。所以接下来，就重点介绍一下经典Benders割和基于逻辑的Benders割的区别。

在网络上关于经典Benders割推导的解释有很多角度，我本身数学基础比较薄弱，很多解释角度我看不懂，但这个回答给出的解释角度我认为是相对好理解的。他这里说的意思大概就是——推导经典Benders分解割的过程实际上就是把变量分组，转换为一个双层规划问题，然后利用线性规划对偶理论把内层规划问题转换为约束条件。什么意思？接下来我们按这套思路走一遍经典Benders割的推导。

我们考虑这样一个原问题（P）...，我们想用经典benders分解来求解这个问题，OK，我们对这个问题的变量进行分组，改写成双层规划的形式，假设外层是对变量y进行优化，内层是对变量x进行优化。注意到，我们做这样的改写之后，它的内层是一个线性规划问题，并且这个内层规划的可行域是和参数y有关系的，不同的y内层规划可能会算出来不一样的结果，因此我们把这个内层规划问题的结果记作q(y)。现在，我们使用上一页提到的那个思路，用线性规划

对偶理论把这个内层规划问题转换一下，转成对偶的形式，也就是q(y)= ... 。内层规划问题做对偶转换后我们会发现一个特别好的性质，就是内层规划问题对偶的可行域和参数y，也就是外层规划问题的变量y是没有关系的。所以，我们这里可以做一个假设，这个假设就是说内层规划的对偶始终是有可行解的，这个假设在实际问题中是始终成立的，如果不成立就说明你的模型建错了。因为如果内层规划的对偶没有可行解，根据线性规划对偶理论，我们就知道内层规划一定要么是无可行解，要么是无界解，从而导致原问题（P）也是无可行解或者无界解，这个时候无论你怎么调整外层规划中y的取值，内层规划的解的情况都不会改变，因为内层规划对偶的可行域和y是没有关系的。有了这个假设，我们就可以进行下一步了，思考一下，我们能不能直接在原问题（P）中用内层规划对偶替换掉内层规划呢？答案是不能的，因为直接带进入会导致原问题（P）的可行域发生变化，例如，对于内层规划问题某个y是不可行的，根据线性规划对偶，我们把这个y代入到内层规划的对偶里面去，内层规划问题解的情况可能是无可行解或者无界解对吧，根据我们刚才的假设，内层规划对偶在内层规划无可行解的时候一定是无界解。但是无界解也属于是可行解对吧。也就是说一个对于内层规划问题不可行的y对于内层规划对偶是可行的。为了解决这个问题，我们需要添加约束，让使得内层规划对偶取得无界解的那些y也变成不可行解。这要怎么做到呢？我们需要了解一些凸集相关的内容，请大家看右下角这张图，我们知道线性规划问题的可行域是一个凸多面体，我们取更一般的例子，假设这个凸集不是封闭的。在这个凸集对应的区域会有很多点，这些点中会有一些特殊的点，这些点不能写成其它点的凸组合，比如图上那两个蓝色的点，这样的点我们称他们为这个凸集的极点。在这个凸集对应的区域呢也会有很多的线，我们以左侧那个极点为例，我们从这个极点的位置发射一些光线出去，如果这个光线不能照射到凸集以外的区域，我们就称这样的光线为凸集在那一点的方向，比如图中绿色的光线。在这些绿色光线中还会存在一些特别的光线，这些光线不能被其他绿色光线的非负组合表达出来，我们就说这样的光线是凸集在这一点的极方向，比如图中红色的这两条光线。对于线性规划，可以证明，它最优解一定可以在某个极点处取到，而当目标函数梯度方向和极射线方向是锐角，就会导致无界解的情况。因此，要使得让内层规划对偶无界的y也变成不可行解，我们只需要添加约束，让内层规划对偶的目标函数的梯度方向和它可行域的极射线成为钝角就行了，也就是 ... 。最后一步，为了把双层规划再转换回单层规划，我们再引入辅助变量q，把内层规划对偶转化为最小化问题（这个过程会用到极点）。也就是... 。

经过这一系列操作，我们就得到了原问题（P）的等价形式，也就是... ，这个形式中的第一类约束就是经典benders可行割，而第二类约束则是经典benders最优割，推导结束。接下来的问题就是，我们应该怎么去解这个问题呢？这里就要涉及到行生成算法了。和列生成算法一样，列多了我们就先不考虑所有列，行多了，我也不考虑所有行了，我先考虑一部分行甚至是一行都不考虑，得到下面这个主问题（MP），把刚刚讲的那个内层优化问题叫做子问题（SP），我们直接去解这个主问题（MP），然后把求解结果y代入到子问题（SP）中，看子问题（SP）的求解结果，如果无解我们就提取子问题对偶的极射线构造可行性割加回到主问题（MP）中，如果子问题（SP）有最优解，我们就把他的对偶变量提取出来构造最优割加回主问题（MP），这两步操作求解器都是有提供接口的，如此往复。关于这个算法为什么有效，这里有一个解释就是说现实问题中的约束非常多，成千上万，但是实际上对我们找最优解有帮助的约束可能就几百个。时间关系，这里就不再赘述了。

接下来，我们再来看LBBD。首先是关于推理对偶的内容，这个概念非常难，首先推理对偶的解就非常抽象，在hooker给的定义中，推理对偶解是一个证明，这个解可以是任何形式，它甚至可以是一段话，比方说“如果x1=x2=1，那么原问题一定无解”，或者说“如果x1+x2=0，那么

原问题的最优解目标函数值至少是5”。那这样的结论或者说是证明又是从哪来的呢？来自我们对问题的推理和分析过程，所以你去分析这个问题什么情况会不可行，或者什么情况，下界至少是某个值的这个逻辑推理过程就是在求解这个问题的推理对偶，而你分析结果就是推理对偶的解。我们再把这样的分析结果，也就是推理对偶的解翻译成约束条件就可以得到逻辑benders可行割或者是最优割了。从实践的角度出发，理不理解推理对偶影响不大，生成逻辑benders割的方法就是把对问题什么时候不可行，什么时候下界至少是多少的分析结果用数学约束的形式表达出来。

逻辑benders分解这个需要我们自己对问题进行推理和分析的特点就给我们解决问题带来了很大的麻烦。但某个方面讲这也是一种优势，应用LBBD需要我们根据具体问题具体分析。实际上，也不是所有基于逻辑的benders割都是需要我们从头设计的，有一部分割是别人总结整理好的，我们可以套用模板。比如，下面这些绿颜色的割都是可以套用模板的设计的，但是往往它们的效果不会好于针对具体问题定制的割。

下面来看一个经典例子，看看LBBD到底是咋用的，这个案例来自16年的IJC期刊，是第一个把LBBD应用到求解考虑准备时间的并行机调度问题中的论文。并行机调度问题是非常经典的问题，非常多的问题实际上都可以归结为并行机调度问题，比如说离散情形下的泊位分配，二维装箱问题，VRP问题等等。这个问题就是说我现在有一批等待加工的工件，和几台加工工件的机器，每个工件在不同的机器上的加工和等待时间不一样，这个等待时间指的是上一个工件加工完得等多长时间才能加工下一个工件，我们应该怎么样把这些工件分配到机器上面去，然后再决策各个机器上工件的加工次序，使得最后一个工件的完工时间最短。

这是一个经典的Np-难问题，这是篇文章给出的数学模型，直接求解它是比较困难的。用LBBD处理它是什么思路呢？我们分析一下这个问题，它的决策过程实际上设计到两个环节，第一个环节是分配，第二个环节是调度。那一个自然的思路就是我先分配，然后再根据分配的结果去做调度对吧，这样就把复杂模型简化了。我们把做分配的问题，就是把原问题中跟调度相关的变量y及其约束给松弛掉，称为主问题，就是右上角这个模型。我们把原问题中主问题的分配方案给固定住，并且忽略掉主问题中的变量，就可以得到子问题的模型，在这个问题中子问题一共有|M|个，|M|是机器的个数，相当于是一直各个机器上有那些加工工件，决策它们的加工次序，也就是|M|个单机调度问题。至此，分解过程就结束了，原问题得到了简化。

剩下的就是去设计割了，实际上他这篇文章的割就可以套模板。（3.9）这条约束就是一条针对调度和分配问题的通用分析割模板，hooker在他的书里面已经证明了这条割的有效性了。这个模板是什么意思呢？就是说我在某一次分配把集合J分配到了某个机器上加工，这台机器的最早完工时间是$v^*(J)$，因为单个机器上的最早完工时间一定是所有机器最早完工时间的下界，我们把这个割加到主问题的效果就是说如果主问题你再做出同样的分配，那你的下界至少应该是$v^*(J)$，如果你对这台机器做出的分配不同，比方说移除了工件j，那你的下界最多减少$\alpha_j$。套模板就是需要我们根据具体问题把这个$\alpha_j$给分析出来。在这个问题里面就是去估计把原本分配给机器$i'$的工件j拿走，机器$i'$的完工时间最多降多少，也就是这里的参数$\theta_{hi'j}$，然后把这个参数套到模板里面就可以得到这篇文章的最优割了。这个问题不涉及到设计可行割，因为他的这个子问题始终是可行的。这就是LBBD的大概应用思路。

这篇文章相对来说很古老了，已经是10年前的文章了。目前大家去用LBBD方法会比它这篇文章复杂的多。在做干散货运营的论文中，大家一般都会去做一个叫主问题增强的东西，就是想办法设计一些好的割引入一些子问题信息，提前把主问题中不好的解给删掉。还有些论文会设

计启发式算法对割做增强处理，大致思路就是用贪婪算法把那些不起作用的变量从割里面删掉。这部分内容我还在学习中，以上就是我本次汇报的全部内容，谢谢大家。

---