Document for DLWMCBG

1. Concepts
   1. Reachable Set

在不同语境下，可达集合（Reachable Set）分别代表不同的含义。在Insert x操作时，Reachable Set指的是x可通过交替路径（alternating path）到达的Z集合中元素所组成的子集，它们可以被x替代以形成另一个相同大小的Z。在Insert y操作时，Reachable Set指的是y可通过交替路径到达的中的元素所组成的子集，它们可以与y匹配形成一个更大的Z集合。

1. Insert X
   1. Message Transmission

Let be an internal node, its left child node, and its right child node.

* + 1. Format of the Message

Computation proceeds along a path from a leaf to the root. For each node except the root in the path, a message is sent from a child node to its parent node. The format of the message is

Specifically, it is

in which and respectively denote the inserted and deleted element in , , of the child node.

* + 1. Messages from Left

左侧消息传递策略（待证明）

1. 左侧插入成功
   1. 直接加入到P.ZL与P.Z集合中
2. 左侧插入失败
   1. 左侧Transfer
      1. \_bZ在P.Z中（是否等价于在P.ZL中？）
         1. 使用EE结构求解\_aZ在P.ZL中的Replaceable Set R
         2. 求出R中end最大的元素maxEndInR
         3. 从P.Z和P.\_ZL中删除maxEndInR
         4. maxEndInR加入到P的ES结构中（P->insertXintoESinNode(bz)）
         5. 根据结果更新Message
      2. \_bZ不在P.Z中
         1. 使用EE结构求解\_aZ在P.ZL中的Replaceable Set R（为何可用EE结构求解R？待严格证明）
         2. 求出R中end最大的元素maxEndInR
         3. 若maxEndInR.e<P.rightChild.Y.s（由L中的transfer变为P中左侧的infeasible）
            1. 求出R中w最小的元素，从P.Z与P.ZL中删除，再将该元素进到I中
            2. 更新Message
         4. 若maxEndInR.e>P.rightChild.Y.s（依然transfer）
            1. 在P.Z与P.ZL中删除maxEndInR
            2. maxEndInR加入到P的ES结构中（P->insertXintoESinNode(bz)）
            3. 更新Message
   2. 左侧Infeasible
      1. \_bZ在P.Z中（是否等价于在P.ZL中？）
         1. 在P.Z与P.ZL中删除\_bZ
         2. Message不变
      2. \_bZ不在P.Z中
         1. 使用EE结构求解\_aZ在P.ZL中的Replaceable Set R（为何可用EE结构求解R？）
         2. 求出R中end最大的元素maxEndInR
         3. 若maxEndInR.e<P.rightChild.Y.s
            1. 求出R中w最小的元素，从P.Z与P.ZL中删除，再将该元素进到I中
            2. 更新Message
         4. 若maxEndInR.e>P.rightChild.Y.s（由L中的infeasible变为P中的transfer）
            1. 在P.Z与P.ZL中删除maxEndInR
            2. maxEndInR加入到P的ES结构中（P->insertXintoESinNode(bz)）
            3. 更新Message

后续工作：

1. 整合右侧消息传递内容（整合以后左侧的消息传递策略可能会产生错误，不确定）。
2. 分裂节点时维护相应ZIT集合的优化算法。目前算法时当发生分裂时，清空L与P中的Z、I、T，然后在L中全部重新插入，最后进行一次左侧消息传递。即目前分裂中的代码与左侧消息传递中的代码时一致的。
3. 优化传递算法，消除重复部分。
   * 1. Messages from Right
4. **在R中x直接加入成功**

msg: (, , , , , ) // if (msg.\_aT.\_id == -1 && msg.\_aI.\_id == -1)

在P中加入x，即：P->insertXintoESinNode()

1. **在R结点，x在Z中抢占x’，x’进入T集合**

msg: (, , , , , ) // if (msg.\_aI.\_id == -1 && msg.\_aT.\_id != -1)

* 1. ，即在R中Z没有变化

在P中，直接进入T

* 1. 若

基于P的计算的可替换集合

* + 1. 若

直接抢占，进入T

* + 1. 若

在P中重新加入，即P->insertXintoESinNode()

* 1. 若

在P中重新加入，即P->insertXintoESinNode()

1. **在R结点，x在Z中抢占x’，x’进入I集合**

msg: (, , , , , ) // if (msg.\_aI.\_id != -1 && msg.\_aT.\_id == -1)

* 1. ，即在R中Z没有变化

在P中，直接进入T

* 1. 若

直接抢占，进入I

* 1. 若

在P中重新加入，即P->insertXintoESinNode()

1. Insert Y
   1. Insert Y in an ES Leaf

Assume return the greatest tight point that is less than y, and return if there is no such one. (After y is inserted, there is no tight point after y.)

Algorithm

Input: Z, T, I, y

Output: Z, T, I

1. Let
2. Let
3. Let
4. If ()
5. Let such that
7. else if ()
8. Let such that
9. * 1. Case of
10. Adjust the sets I and T according to the inserted y.
11. If there is such that , select the one that has the maximum weight to augment Z.
    1. Internal Node

Let be an internal node, its left child node, and its right child node.

在P的ES结构中，若该结构时满的，那么加入y时，存在一个其左侧的点和一个其右侧的点。那么所有或的节点都不在可替换集合中。

* + 1. Format of the Message

Same as the last section

* + 1. Messages from Left

In P, let be the compensable set of , be the compensable set of w.r.t. , be the compensable set of w.r.t. .

**Problem 1**: &&

Description: In this case, y is inserted in and include some in which are matched in . Let and be two such vertices. If they are removed from , there must be some vertices in to be compensated since . Let and be the compensable sets w.r.t. and , respectively. Then it is necessary to find the one with the maximum weight in .

Question to Hypothesis: Is the compensable set an interval in unmatched sorted set?

Idea to be verified:

1. When inserting y from a left child, the reference set should be , not .
2. Only when there is no compensable in , consider .

Algorithm

1. Let
2. Let
3. Let
4. Let such that has the least
5. If ()
6. Let
7. Let
8. or Let
9. else
10. Let
11. if ()
12. Let such that has the maximum weight
13. if ()

16. else
18. else if ()
19. Let such that has the least
20. if () //

23. else

**Problem 2**: 当将某个拉回到时，右侧的可替换集合

* + 1. Messages from Right

Assume , applied in an equal-start CBG, returns the greatest tight point that is less than , and returns if there is no such one.

Algorithm

1. Let
2. Let
3. Let
4. if ()
5. Let such that
6. if ()
8. else
10. else if ()
11. Let such that
12. if ()
14. else if ()

Notes: The sets are maintained implicitly in code.