Optimal Greedy Method for Generalized Activity Selection

Chee Yap
Department of Computer Science, Courant Institute, New York University, USA

Bingwei Zhang[1]
Department of Computer Science, Courant Institute, New York University, USA

Abstract

The generalized activity selection problem is this: given $m \geqslant 1$ and a set A of intervals representing time spans of activities, we want to select m subsets $\{C_i \subseteq A : i = 1, \ldots, m\}$ such that the intervals in each C_i are pairwise disjoint, and $|\bigcup_{i=1}^m C_i|$ is maximized. The well-known activity selection problem corresponds to $m = 1$. We provide an $O(n \log n)$ greedy algorithm. Proving its optimality is more subtle than in the $m = 1$ case.

2012 ACM Subject Classification Replace ccsdesc macro with valid one

Keywords and phrases Dummy keyword

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Funding This work is funded by NSF Grant #CCF-2212462.

1 Introduction

The **activity selection problem** is used in standard text books such as [1, 4] to illustrate the greedy method.[2] The problem is this: given a set A of intervals, compute a compatible set $C \subset A$ of maximal size. Here, C is **compatible** if the intervals in C are pairwise disjoint. Each interval $I \in A$ represents the time span of an activity. In the following, we assume half-open intervals of the form $I = (s, f]$ or $I = (s(I), f(I)]$ where $s(I)$ and $f(I)$ are the start and finish time of activity I. So the set $\{I, J\}$ is compatible iff $f(I) \leqslant s(J)$.

The problem is usually attributed to Gavril (1972) who showed that the more general problem of maximal independent set in a chordal graph [3, §3] can be computed in $O(n^3)$ time, given the perfect elimination order[3] of G. When specialized to interval graphs, the complexity improves to $O(n \log n)$ [1, §15.1].

In this paper, we consider the following generalization: given $m \geqslant 1$ and A, compute m compatible sets $C_1, \cdots, C_m$ of A such that $|C_1 \cup \cdots \cup C_m|$ has maximal size. Note that wlog, we may assume the C_i's are pairwise disjoint, and allow some C_i's to be empty. We may call this the **multiroom activity selection problem** because we imagine the activities in C_i to be assigned to the ith room. If m is fixed, we speak of the **m-room activity selection problem**. The original problem of Gavril is the 1-room case. Our main result is a greedy algorithm to compute an optimal solution for this problem. We show that its complexity remains $O(n \log n)$, but its correctness is considerably more subtle than the 1-room case.

Related Problems: Despite it's naturalness, our generalization appears to be new. There is a known generalization to the weighted case [5, 1, 4]: suppose we are given a function $W : A \to \mathbb{R}_{>0}$, and the goal is to compute a compatible set $C \subseteq A$ whose weight

---

[1] Optional footnote, e.g. to mark corresponding author
[2] It is also popular in coding websites such as https://www.geeksforgeeks.org/.
[3] In her original paper, she gave an $O(n^4)$ method to compute the elimination order. Subsequently Rose, Luecker and Tarjan [6]. gave an $O(m + n)$ algorithm based on BFS. In this paper, the parameters n, m denotes the number of vertices and edges of a graph G.

42nd Conference on Very Important Topics (CVIT 2016).
Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:14
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

$W(C) = \sum_{I \in C} W(I)$ is maximum. Unfortunately, this generalization needs a dynamic programming solution with complexity $\Omega(n^2)$; in contrast to our generalization is still amenable to a $O(n \log n)$ greedy approach.

As noted above, Gavril viewed the activity selection problem as an maximum independent set problem. More precisely, the set of intervals A defines an interval graph $G(A)$ with A as vertex set and edges are pairs $\{I, J\}$ of intervals with non-empty intersection. So a compatible set $C \subseteq A$ is just an independent set of $G(A)$. In this graph setting, Gavril considered the **minimum coloration problem** [3, §2]. In modern terminology, this is computing the **chromatic number** $\chi(G)$ of a graph G, She gave an $O(n^2)$ algorithm to compute $\chi(G)$ for a chordal graph G. When G is an interval graph, Kleinberg and Tardos ([4, p. 122], [1, Ex. 16.1-3, p. 179]) improved it to $O(n \log n)$ solution. The topic of scheduling has many similarities with activities selection. For instance, in the job-shop scheduling problem we are given n jobs and m machines (like our intervals and rooms). But our interval is replaced by a job J_i that is characterized only by its duration $\mu(J_i) > 0$: the algorithm not only assigns J_i to a machine, but it also has to schedule the starting time $s(J_i)$.

Finally, there is some geometric content inherent in interval graphs, and it will be useful to adopt the notion of "stabbing numbers" [2, §10.4, p.227] from computational geometry. Consider the following **interval stabbing problem**: *given $m \geqslant 1$ and A, find a subset $C \subseteq A$ of maximum size and with stabbing number $\leqslant m$.* Here, the **stabbing number** of C is defined as

$$stab\#(C) := \max_{x \in \mathbb{R}} |x \wedge C|$$

where $x \wedge C := \{I \in C : x \in I\}$ is the set of intervals of C that are "stabbed" by x. Let $Opt(A, m)$ denote the size $|C|$ of the optimal solution.

▸ **Observation 1.** *A set C of intervals can be partitioned into k non-empty compatible sets iff $stab\#(C) = k$.*

This will follow from the correctness of our greedy algorithm below. Therefore, the interval stabbing problem is equivalent the multiroom activity selection problem.

2 Optimal Greedy Algorithm for 2-Room Activity Selection

We first give an optimal solution for the two room case ($m = 2$). It will make the transition of the general case much easier.

Here is a brief overview of the algorithm: let the input set A of n intervals be sorted by their finish times:

$$I_1 <_f I_2 <_f \cdots <_f I_n \qquad (1)$$

where $I <_f J$ iff $f(I) < f(J)$. Note that we assume that the finish times are distinct. This is without loss of generality since we can break ties arbitrarily. Likewise, assume that each $s(I_i) \geqslant 0$. We maintain two lists $Room[j]$ ($j = 1, 2$) holding compatible sets of intervals. For interval I_i ($1 = 1, \ldots, n$), we process I_i by either Accepting or Rejecting it. This is usual Accept/Reject paradigm of greedy methods. The twist is that, we accept I_i by appending it to either $Room[1]$ or $Room[2]$.

The critical question is how to process an interval. For this purpose, let $fTime(j)$ be the finish time of the last interval placed into $Room[j]$ ($j = 1, 2$). So $fTime(j)$ is increased (in view of (1)) whenever we place a new interval into $Room[j]$. Introduce a variable $front$ whose value is 1 iff $fTime(1) \geqslant fTime(2)$; otherwise $front = 2$. Also maintain a complementary variable *back* satisfying the invariant $back = 3 - front$. Call $Room[front]$ and $Room[back]$ the **front** and **back room** respectively. The ordered pair

$$(fTime(back), fTime(front))$$

is called the **state** of the rooms. So if $\boldsymbol{s} = (a, b)$ is a state, then $a \leqslant b$. The **initial state** is $\boldsymbol{s}_0 := (0, 0)$. It turns out that for all non-initial states, we always have the strict inequality $a < b$. In this case, we prefer to denote the state in a more distinctive way, "$(a < b)$". Suppose that the current state is $\boldsymbol{s} = (a \leqslant b)$. We say $I = (s, f]$ is **applicable** to $\boldsymbol{s} = (a < b)$ if $b < f$. If I is applicable to $\boldsymbol{s} = (a < b)$, then we can apply I to $\boldsymbol{s}$ to transform it to a new state $\boldsymbol{t} = (a' < b')$ defined as follows:

$$(a' < b') := \begin{cases} (a < f) & \text{if } b \leqslant s & \text{.......Case(I): keep,} \\ (b < f) & \text{if } a \leqslant s < b & \text{.......Case(II): flip,} \\ (a < b) & \text{if } s < a & \text{.......Case(III): reject.} \end{cases} \tag{2}$$

Write $(a < b) \xrightarrow{I} (a' < b')$ to indicate this transformation. So the sequence (1) induces a sequence of transformations:

$$\boldsymbol{s}_0 \xrightarrow{I_1} \boldsymbol{s}_1 \xrightarrow{I_2} \cdots \xrightarrow{I_n} \boldsymbol{s}_n. \tag{3}$$

This sequence is well-defined because each I_i is applicable to the previous state $\boldsymbol{s}_{i-1}$.

Running Example. Suppose the sorted input are these 6 intervals:

$$Sort(A) = (I_1(0,3] <_f I_2(1,4] <_f I_3(4,6] <_f I_4(2,7] <_f I_5(6,8] <_f I_6(4,9]). \tag{4}$$

This induces the following state transformations:

$$\boldsymbol{s}_0(0,0) \xrightarrow[keep]{I_1} \boldsymbol{s}_1(0<3) \xrightarrow[flip]{I_2} \boldsymbol{s}_2(3<4) \xrightarrow[keep]{I_3} \boldsymbol{s}_3(3<6) \xrightarrow[rej]{I_4} \boldsymbol{s}_4(3<6) \xrightarrow[keep]{I_5} \boldsymbol{s}_5(3<8) \xrightarrow[flip]{I_6} \boldsymbol{s}_6(8<9). \tag{5}$$

Note that we also indicate the case (keep/flip/reject) of each transformation.

∎

We have now completely described our algorithm which is called $Greedy2(A)$ here:

Greedy2(A) $\to B$
INPUT: A is a set of n intervals
OUTPUT: Set $B \subseteq A$ with stabbing number $\leqslant 2$ with maximal cardinality.
Sort the n intervals of A as in (1).
Let $Room[1], Room[2]$ be compatible sets of intervals, initially empty.
Let $fTime(1) \leftarrow fTime(2) \leftarrow 0$
and $front \leftarrow 1$, $back \leftarrow 2$.
For $i = 1, \ldots, n$,
Case(I) If $s(I_i) \geqslant fTime(front)$,
$Room[front].append(I_i)$
$fTime(front) \leftarrow f(I_i)$ $\lhd$ *keep the value of front*
Case(II) Else if $(s(I_i) \geqslant fTime(back)$,
$front \leftrightarrow back$ $\lhd$ *flip front and back*
$Room[front].append(I_i)$
$fTime(front) \leftarrow f(I_i)$
Case(III) Else $\lhd$ $s(I_i) < fTime(back)$
Reject I_i $\lhd$ *i.e., do nothing*
Return $B = Room[1] \cup Room[2]$

Note that Cases (I-II) in the algorithm implement rule (2). Moreover, we identify state $\boldsymbol{s}_i$ ($i = 1, \ldots, n$) in (3) as the state at the end of the ith iteration of the for-loop. We also see why Case(I) and Case(II) are called "keep" and "flip" cases. In the running example above, $Greedy2(A)$ returns the set $Room[1] \cup Room[2]$ where

$$Room[1] = (I_1(0,3], I_6(4,9]), \qquad Room[2] = (I_2(1,4], I_3(4,6], I_5(6,8].)$$

Only one interval $I_4(2,7]$ is rejected.

The correctness of $Greedy2$ is based on the following theorem:

▶ **Theorem 2** (Key).
(a) *If $stab\#(A) \leqslant 2$, then $Greedy2(A) = A$.*
(b) *If $B \subseteq A$ then $|Greedy2(B)| \leqslant |Greedy2(A)|$.*

▶ **Corollary 3.** *Greedy2 is correct,*
i.e., if $B = Greedy2(A)$ then $stab\#(B) \leqslant 2$ and $|B| = Opt(A, 2)$.

Proof of Corollary If $B = Greedy2(A)$ then clearly $stab\#(B) \leqslant 2$ because B is the union of two compatible subsets of A. This implies that $Opt(A, 2) \geqslant |B|$. So it remains to prove that $Opt(A, 2) \leqslant |B|$. By definition of $Opt(A, 2)$, there is a set $B^* \subseteq A$ such that $|B^*| = Opt(A, 2)$ and $stab\#(B^*) \leqslant 2$. Thus

$$\begin{aligned} Opt(A,2) &= |B^*| && \text{(by choice of } B^*) \\ &= |Greedy2(B^*)| && \text{(by Theorem 2(a))} \\ &\leqslant |Greedy2(A)| && \text{(by Theorem 2(b))} \\ &= |B| && \text{(by choice of } B). \end{aligned}$$

Q.E.D.

Of the two parts in Theorem 2, part(a) is the easier to show:

Proof of Theorem 2(a) Assuming $stab\#(A) \leqslant 2$, we must show that $Greedy2(A) = A$, i.e., no interval of A is rejected. Let $s_i = (a_i < b_i)$ $(i = 1, \ldots, n)$ be the states in (3). Our result follows from 3 CLAIMS:

CLAIM 0: If $a_i = 0$ then I_i cannot be rejected. Pf: $a_i = 0$ means that $Room[back]$ is empty. In this case, the transformation $s_{i-1} \xrightarrow{I_i} s_i$ falls under Case(I) or (II), i.e., I_i is not rejected.

CLAIM 1: If $a_i > 0$ then there are some $j < k \leqslant i$ such that

$$a_i \in I_j \cap I_k \tag{6}$$

Pf: Note that $b_i = f(I_i)$ and $a_i = f(I_j)$ for some $j < i$. Suppose I_j is in $Room[1]$ (the other case is similarly argued). That means that at the end of the jth iteration, $front = 1$. Since I_i must be in $Room[2]$, there is a smallest k $(j < k \leqslant i)$ such that I_k caused a flip (Case(II)), i.e., $s(I_{k-1}) < a_i < f(I_k)$. Thus $a_i \in I_j \cap I_k$, as CLAIMed.

CLAIM 2: If $a_i > 0$ then $a_{i+1} > 0$ and I_{i+1} is never rejected. Pf: Consider the transition $s_i \xrightarrow{I_{i+1}} s_{i+1}$. This cannot result in Case(III) (the rejection of I_{i+1}) because it would imply that $a_i \in I_{i+1}$, and (combined with CLAIM 1) implies $stab\#(a_i, A) = 3$, contradicting the assumption $stab\#(A) \leqslant 2$. Furthermore $a_{i+1} > 0$ holds because Case(I) implies $a_i = a_{i+1}$ and Case(II) implies $a_{i+1} \in I_i \cap I_{i+1}$. **Q.E.D.**

2.1 Setup for the proof of Theorem 2(b)

Theorem 2(b) clearly follows if we prove that

$$|Greedy2(A)| \leqslant |Greedy2(A^+)| \tag{7}$$

for all A and $A^+ = A \cup \{I^+\}$. To simplify the setting, note that (7) holds if I^+ is rejected by $Greedy2(A^+)$. Therefore we may assume I^+ is accepted by $Greedy2(A^+)$. Let

$$Sort(A^+) = (J_1 <_f J_2 <_f \cdots <_f J_{n+1}). \tag{8}$$

Thus I^+ appears in (8) as J_{i^+} for some $1 \leqslant i^+ \leqslant n+1$. The sequence (8) induces these state transformations

$$\boldsymbol{s}_0 \xrightarrow{J_1} \boldsymbol{s}_1 \xrightarrow{J_2} \cdots \xrightarrow{J_{n+1}} \boldsymbol{s}_{n+1}.$$

We want to compare the state transformations of $Greedy2(A^+)$ with those of $Greedy2(A)$. But $Greedy2(A)$ produces one less state than $Greedy(A^+)$. We will make them agree by an artifice: let $A^0 := A \cup \{I^0\}$ where $I^0 := (-1, f(I^+)]$ be an artificial interval that will always be rejected by our algorithm. Therefore the sorted sequence $Sort(A^0)$ agrees with (8) except that $J_{i^+} = I^+$ is replaced by I^0. Let $(\boldsymbol{t}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_{n+1})$ be the corresponding sequence of states induced by $Sort(A^0)$. Now we can compare $\boldsymbol{s}_i$ with $\boldsymbol{t}_i$ for $i = 1, \ldots, n+1$. Clearly, $\boldsymbol{s}_i = \boldsymbol{t}_i$ for $i < i^+$. But $\boldsymbol{s}_{i^+} \neq \boldsymbol{t}_{i^+}$ since I^+ is accepted and I^0 is rejected. Let $P_i := \left[\begin{smallmatrix}\boldsymbol{s}_i\\ \boldsymbol{t}_i\end{smallmatrix}\right]$ be the ith **state pair** (or simply **pair**). Also $\boldsymbol{s}_i$ and $\boldsymbol{t}_i$ are the **upper** and **lower states** of P_i. We call P_i an **equality pair** if $\boldsymbol{s}_i = \boldsymbol{t}_i$. So the first inequality pair is P_{i^+} called the **critical pair**. In the rest of our analysis, we consider this sequence of state pairs

$$P_0 \longrightarrow P_1 \longrightarrow \cdots \longrightarrow P_n \longrightarrow P_{n+1} \tag{9}$$

Moreover, for $i \neq i+1$, we may write

$$P_{i-1} \xrightarrow{J_i} P_i$$

since the same I_i is used to transform the upper and lower states of P_{i-1}.

Running Example (contd). Let $I^+ = (3,5]$, then $Sort(A^+)/Sort(A^0)$ is

$$(J_1(0,3] <_f J_2(1,4] <_f (I^+/I^0) <_f J_4(4,6] <_f J_5(2,7] <_f J_6(6,8] <_f J_7(4,9]) \quad (10)$$

It induces the transformations of pairs:

$$P_0 \xrightarrow[kk]{J_1} P_1 \left[\begin{smallmatrix} 0<3 \\ 0<3 \end{smallmatrix}\right] \xrightarrow[ff]{J_2} P_2 \left[\begin{smallmatrix} 3<4 \\ 3<4 \end{smallmatrix}\right] \xrightarrow[fr]{I^+/I^0} P_3 \left[\begin{smallmatrix} 4<5 \\ 3<4 \end{smallmatrix}\right] \xrightarrow[fk]{J_4} P_4 \left[\begin{smallmatrix} 5<6 \\ 3<6 \end{smallmatrix}\right]$$
$$\xrightarrow[rr]{J_5} P_5 \left[\begin{smallmatrix} 5<6 \\ 3<6 \end{smallmatrix}\right] \xrightarrow[kk]{J_6} P_6 \left[\begin{smallmatrix} 5<8 \\ 3<8 \end{smallmatrix}\right] \xrightarrow[rf]{J_7} P_7 \left[\begin{smallmatrix} 5<8 \\ 8<9 \end{smallmatrix}\right]. \quad (11)$$

Since $i^+ = 3$, the critical pair in (11) is $P_3 = \left[\begin{smallmatrix} 4<5 \\ 3<4 \end{smallmatrix}\right]$. Note that the subscript $xy \in \{k, f, r\}^2$ in the notation

$$P_{i-1} \left[\begin{smallmatrix} \boldsymbol{s}_{i-1} \\ \boldsymbol{t}_{i-1} \end{smallmatrix}\right] \xrightarrow[xy]{J_i} P_i \left[\begin{smallmatrix} \boldsymbol{s}_i \\ \boldsymbol{t}_i \end{smallmatrix}\right] \quad (12)$$

says that the upper transformation $\boldsymbol{s}_{i-1} \to \boldsymbol{s}_i$ is Case x and lower transformation $\boldsymbol{t}_{i-1} \to \boldsymbol{t}_i$ is Case y. Call xy the **type** of the transition. The type is completely determined by P_{i-1} and P_i. Check: the lower transformations in (11) is basically given by (5).

■

Equivalent Pairs. For $P = \left[\begin{smallmatrix} a \leqslant b \\ c \leqslant d \end{smallmatrix}\right]$, let $\text{supp}(P) := \{a, b, c, d\}$. E.g., $\text{supp}(\left[\begin{smallmatrix} 1<4 \\ 0<4 \end{smallmatrix}\right]) = \{0, 1, 4\}$. We say two pairs P and Q are **equivalent** if there is a monotone function $T : \text{supp}(P) \to \text{supp}(Q)$ such that $Q = T(P) = \left[\begin{smallmatrix} T(a) \leqslant T(b) \\ T(c) \leqslant T(d) \end{smallmatrix}\right]$. Here, T is a **monotone** means $x \leqslant y$ iff $T(x) \leqslant T(y)$. It is easy to check that this is an equivalence relation on state pairs, denoted $P \equiv Q$. E.g., , $\left[\begin{smallmatrix} 1<4 \\ 0<4 \end{smallmatrix}\right] \equiv \left[\begin{smallmatrix} 3<5 \\ 1<5 \end{smallmatrix}\right]$ but $\left[\begin{smallmatrix} 0<3 \\ 0<2 \end{smallmatrix}\right] \not\equiv \left[\begin{smallmatrix} 0<2 \\ 0<3 \end{smallmatrix}\right]$. Call each equivalence class a **class pair**.

Strict Pairs. A state $\boldsymbol{s} = (a, b)$ is **strict** if $a < b$. Other than the initial state $(0, 0)$, subsequent states must be strict. A pair $P = \left[\begin{smallmatrix} \boldsymbol{s} \\ \boldsymbol{t} \end{smallmatrix}\right]$ is **strict** if both $\boldsymbol{s}$ and $\boldsymbol{t}$ are strict. It is easy to verify that for all $i \geqslant 2$, $P_i = \left[\begin{smallmatrix} \boldsymbol{s}_i \\ \boldsymbol{t}_i \end{smallmatrix}\right]$ is strict. We largely focus on strict pairs; henceforth the unqualified "pair" will mean "strict pair".

Our rule (2) for transforming states easily implies:

▸ **Lemma 4.** *We have $|\text{supp}(P_i)| < 4$ for all P_i in the sequence (9).*

E.g., $P = \left[\begin{smallmatrix} 1<3 \\ 2<4 \end{smallmatrix}\right]$ cannot occur in (9). In view of this lemma, we see that every strict pair is equivalent to one of these 7 **canonical pairs** with support in $\{1, 2, 3\}$:

$$\boxed{\begin{array}{llll} [A] = \left[\begin{smallmatrix} 1<3 \\ 1<2 \end{smallmatrix}\right] & [B] = \left[\begin{smallmatrix} 2<3 \\ 1<2 \end{smallmatrix}\right] & [C] = \left[\begin{smallmatrix} 2<3 \\ 1<3 \end{smallmatrix}\right] & [D] = \left[\begin{smallmatrix} 1<2 \\ 1<3 \end{smallmatrix}\right] \\ \\ [E] = \left[\begin{smallmatrix} 1<2 \\ 1<2 \end{smallmatrix}\right] & [F] = \left[\begin{smallmatrix} 1<3 \\ 2<3 \end{smallmatrix}\right] & [G] = \left[\begin{smallmatrix} 1<2 \\ 2<3 \end{smallmatrix}\right] & \end{array}} \quad (13)$$

Let $V_2 = \{[A], [B], [C], [D], [E], [F], [G]\}$ be the set of canonical pairs from (13). We interchangeably view $\alpha \in V_2$ as a pair as well as an equivalence class. Thus write both $P \equiv \alpha$ and $P \in \alpha$.

▸ **Observation 5.**

(a) *There are exactly 7 class pairs, and they may be identified with the elements of* V_2*.*
(b) *The critical pair* P_{i+} *in (9) is equivalent to either* $[A]$ *or* $[B]$*.*

Transition graph G_2**:** Let $G_2 = (V_2, E_2)$ be the digraph whose edges are defined as follows: $(\alpha \longrightarrow \beta) \in E_2$ iff there exist state pairs $P \equiv \alpha$ and $Q \equiv \beta$ and type $\tau \in \{k, f, r\}^2 \setminus \{rr\}$ such that $P \xrightarrow[\tau]{I} Q$ for some I. We also write $\alpha \xrightarrow[\tau]{} \beta$ in this case, and say[4] that τ is **applicable** to α. E.g. we see that $\tau = kk$ is applicable to every $\alpha \in V_2$ because $P \xrightarrow[kk]{I} Q$ if $s(I) > a$ for all $a \in \mathrm{supp}(P)$. The requirement that $\tau \neq rr$ means that I is accepted by either upper or lower state of P. If $\alpha \xrightarrow[\tau_i]{} \beta_i$ for $i = 1, 2, \ldots$, we can write

$$\alpha \xrightarrow[\tau_1/\tau_2/\cdots]{} \beta_1/\beta_2/\ldots.$$

E.g., We see that there exactly two types, kk and ff, that are applicable to $[E]$; moreover

$$[E] \xrightarrow[kk/ff]{} [E]/[E].$$

The following lemma shows that for $\alpha \neq [E]$, there are exactly 3 types that are applicable to α. It is proved by simple enumeration:

▸ **Lemma 6.**
The complete list of types applicable to each $\alpha \in V_2$ *are enumerated as follows:*
(a) $[A] \xrightarrow[kk/fk/ff]{} [E]/[C]/[C]$
(b) $[B] \xrightarrow[kk/fk/rf]{} [C]/[C]/[D]$
(c) $[C] \xrightarrow[kk/ff/rf]{} [C]/[E]/[G]$
(d) $[D] \xrightarrow[kk/kf/ff]{} [E]/[F]/[F]$
(e) $[E] \xrightarrow[kk/ff]{} [E]/[E]$
(f) $[F] \xrightarrow[kk/ff/fr]{} [F]/[E]/[B]$
(g) $[G] \xrightarrow[kk/kf/fr]{} [F]/[F]/[A]$

These edges completely determine the graph G_2 *as shown in Figure 1.*

Structure of G_2: Node $[E]$ is the only sink. The elementary cycles of G_2 consist of 3 self-loops at the vertices $[C], [E], [F]$, and 3 non-trivial cycles

$$\langle [A] \longrightarrow [C] \longrightarrow [G] \rangle, \quad \langle [B] \longrightarrow [D] \longrightarrow [F] \rangle, \quad \langle [B] \longrightarrow [C] \longrightarrow [G] \longrightarrow [F] \rangle.$$

Define $\Delta : V_2 \to \{1, 0, -1\}$ where

$$\Delta(\alpha) = \begin{cases} 1 & \text{if } \alpha \in \{[A],[B],[C]\} \quad \text{....(positive class)} \\ 0 & \text{if } \alpha \in \{[D],[F],[G]\} \quad \text{....(neutral class)} \\ \frac{1}{2} & \text{else.} \quad \text{....(ambiguous class)} \end{cases} \tag{14}$$

Thus, the classes in V_2 are classified as positive, neutral or ambiguous by Δ. We also call a pair P positive, neutral or ambiguous if its equivalence class is positive, neutral or ambiguous. We will next see how this classification is used.

[4] Unlike $P \longrightarrow Q$ having a unique type, $\alpha \longrightarrow \beta$ may have more than one type.

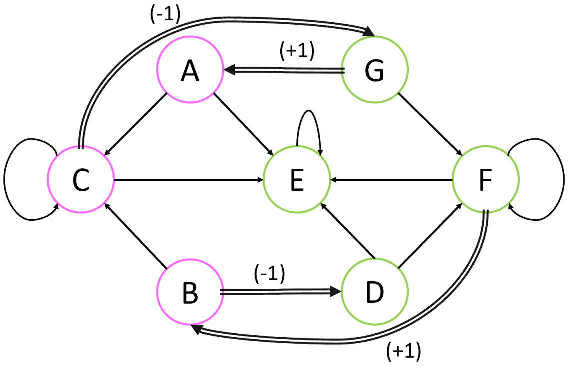

Figure 1 Transition graph G_2: Edges have $-1/0/+1$ weights: they have weight 0 unless noted otherwise.

Proof of Theorem 2(b)

Consider the sequence (9) of pairs, and let

$$\alpha_0 \longrightarrow \alpha_1 \longrightarrow \cdots \longrightarrow \alpha_{n+1} \tag{15}$$

be the corresponding sequence of classes where $P_i \equiv \alpha_i$. Let $U(i)$ (resp., $L(i)$) be the total number of intervals accepted by the upper (resp., lower) states of $P_0, P_1, \ldots, P_i$. Let $\delta(i) := U(i) - L(i)$. Clearly, $\delta(i) = 0$ for all $i < i^*$ and $\delta(i^+) = 1$.

- CLAIM 1: For $i > i^+$, if $P_{i-1} \underset{xy}{\longrightarrow} P_i$ then

$$\delta(i) - \delta(i-1) = W(P \underset{xy}{\longrightarrow} P_i) := \begin{cases} -1 & \text{if } x = r, \\ +1 & \text{if } y = r, \\ 0 & \text{else.} \end{cases} \tag{16}$$

Pf: We verify one of these 3 cases: $x = r$ implies that $U(i) = U(i-1)$. But it also implies that $y \neq r$, and hence $L(i) = L(i-1) + 1$. So

$$\delta(i) = U(i) - L(i) = U(i-1) - (L(i-1) + 1) = \delta(i-1) - 1.$$

The other two cases are similar. This establishes CLAIM 1.

- As defined, $\delta(i)$ depends on the pairs P_j for all $j = 0, \ldots, i$. We next show that $\delta(i)$ depends only on the equivalence class of P_i alone (although $\delta(i)$ can be 0 or 1 if P_i is ambiguous).
- CLAIM 2: The formula for $W(P \to Q)$ in (16) is a function of the underlying classes, i.e.,

$$W(P \to Q) = \Delta(\beta) - \Delta(\alpha) \tag{17}$$

where $P \equiv \alpha$ and $Q \equiv \beta$, and Q is not ambiguous. Pf: Suppose $W(P \to Q) = -1$. From Lemma 6(b,c), we see the two possibilities: $(\alpha, \beta) = ([B], [D])$, or $(\alpha, \beta) = ([C], [G])$. Since $\Delta([B]) = \Delta([C]) = 1$ and and $\Delta([D]) = \Delta([G]) = 0$, we verify (17).
Suppose $W(P \to Q) = +1$. Again Lemma 6(f,g) shows the two possibilities: $(\alpha, \beta) = ([F], [B])$, or $(\alpha, \beta) = ([G], [A])$. We may again verify (17).
Finally, if $W(P \to Q) = 0$, we see that $\Delta(\alpha) = \Delta(\beta)$ in the remaining edges, verifying (17).

- CLAIM 3: $\delta(i^+) = \Delta(\alpha_{i^+}) = 1$.
  Pf: By our setup, $\delta(i^+) = 1$. By Observation 5, P_{i^+} is equivalent to $[A]$ or $[B]$ and $\Delta([A]) = \Delta([B]) = 1$.
- CLAIM 4: For $i \geqslant i^+$,

$$\delta(i) \begin{cases} = \Delta(\alpha_i) & \text{if } \alpha_i \neq [E], \\ \in \{0,1\} & \text{if } \alpha_i = [E]. \end{cases} \tag{18}$$

Pf: We prove this by induction on i. The basis $i = i^+$ is shown in CLAIM 3. If $\alpha_i \neq [E]$ then:

$$\begin{aligned} \delta(i) &= \delta(i-1) + W(P_{i-1} \to P_i) && \text{(CLAIM 1)} \\ &= \Delta(\alpha_{i-1}) + W(P_{i-1} \to P_i) && \text{(induction hypothesis)} \\ &= \Delta(\alpha_{i-1}) + (\Delta(\alpha_i) - \Delta(\alpha_{i-1}) && \text{(CLAIM 2)} \\ &= \Delta(\alpha_i). \end{aligned}$$

Suppose $\alpha_i = [E]$. Since $[E]$ is a sink and $\alpha_{i^+} \neq [E]$, there is a last time j such that $j < i$ and $\alpha_j \neq [E]$. We had proved by induction that $\delta(j) = \Delta(\alpha_j)$. So it remains to show that for all $k > j$, $\delta(k) = \delta(j)$. Looking at Lemma 6(a,c,d,f), we see verify ' that $W(P_j \to P_{j+1}) = 0$ for any transition from an non-ambiguous class into the ambiguous class. This means that $\delta(j+1) = \delta(j)$. Next, we also see that $W(P \to Q) = 0$ for transitions between two ambiguous pairs (by Lemma 6(e)). This proves that $\delta(i) = \delta(i-1) + W(P_{i-1} \to P_i) = \delta(i-1)$. Repeating this, we see that $\delta(i) = \delta(j)$. Our CLAIM follows since $\Delta(\alpha_j) \in \{0,1\}$.

To conclude our proof, CLAIM 4 implies that $\delta(n+1) \in \{0,1\}$. But $\delta(n+1) = |Greedy2(A^+)| - |Greedy2(A^0)|$. Thus $|Greedy2(A^+)| \geqslant |Greedy2(A^0)|$, proving Theorem 2(b).

3 General Case

We now consider the general case of $m \geqslant 2$ rooms. Although many concepts introduced for $m = 2$ remain intact, we need to generalize some.

Our algorithm now maintains m rooms, and the ith interval I_i is again rejected or else it is accepted into one of the m rooms. The ith room is associated with $fTime(i)$, which equals $f(I)$ where I was last put into the room. We have an array $front[1..m]$ where $front[i] \in \{1, \ldots, m\}$ are used to maintain this invariant:

$$fTime[front[1]] \leqslant fTime[front[2]] \leqslant \cdots \leqslant fTime[front[m]]. \tag{19}$$

These inequalities are strict unless both values are 0.

States. We now define a **state** to be $\boldsymbol{s} = (a_1 < a_2 < \cdots < a_k)$ with $1 \leqslant k \leqslant m$ with $\text{supp}(\boldsymbol{s}) = \{a_1, \ldots, a_k\}$. The initial state is (0) where $k = 1$ and $a_1 = 0$; but thereafter, $a_1 > 0$. Note that our notion of states departs slightly from the $m = 2$ case. The **rank** of $\boldsymbol{s}$ is k, corresponding to the number of non-empty rooms so far. We map the sequence $\boldsymbol{f} = (f_1 \leqslant f_2 \cdots\cdots \leqslant f_m)$ of (19) into a state by removing any $f_i = 0$. E.g., $\boldsymbol{f} = (0,0,2,5,6) \mapsto \boldsymbol{s} = (2,5,6)$. The rule (2) for transforming states will now explicitly allow for the increase in rank: if interval I is **applicable** to $\boldsymbol{s} = (a_1 < \cdots < a_k)$ (i.e., $f(I) > a_k$), then its application to $\boldsymbol{s}$ results in a new state $\boldsymbol{t} = (b_1 < \cdots < b_\ell)$ defined as follows: let $i^* := \text{argmax}_{i=0}^{k} \{a_i : a_i \leqslant s(I)\}$ (where $a_0 = 0$).

$$\boldsymbol{t} = \begin{cases} (a_1 < \cdots < \widehat{a_{i*}} < \cdots < a_k < f(I)) & \text{if } i^* \geqslant 1, & \text{.... Case}(i^*) \\ (a_1 < \cdots < \cdots < a_k < f(I)) & \text{if } i^* = 0 \text{ and } k < m, & \text{.... Case}(0) \\ \boldsymbol{s} & \text{if } i^* = 0 \text{ and } k < m, & \text{.... Case}(-1) \end{cases} \tag{20}$$

where the notation $\widehat{a_{i*}}$ in Case(i^*) means that a_{i*} is omitted from the sequence. As usual, we write $\boldsymbol{s} \xrightarrow{I} \boldsymbol{t}$. E.g., let $m = 4$. Then $\boldsymbol{s} = (1,3,5) \xrightarrow{(0,7]} \boldsymbol{t} = (1,3,5,7)$ with $i^* = 0$. Also $\boldsymbol{s} = (1,3,5) \xrightarrow{(4,7]} \boldsymbol{t} = (1,5,7)$ with $i^* = 2$. The **type** of the transition $\boldsymbol{s} \xrightarrow{I} \boldsymbol{t}$ in (20) is **reject** (rej) in Case(-1), **accept** (acc) in Case($i \geqslant 1$) and **extend** (ext) in Case(0). Thus, the rank of $\boldsymbol{s}$ is incremented iff the type is ext. Also $\boldsymbol{s}$ accepts I means that some $a \in \boldsymbol{s}$ is **displaced** by $f(I)$. Write $\boldsymbol{s} \xrightarrow[\tau]{} \boldsymbol{t}$ when the type is $\tau \in \{acc, ext, rej\}$.

▸ **Lemma 7.** *Let $\boldsymbol{s} = (a_1 < \cdots < a_k)$ be a state. If $R \subseteq A$ is the current set of intervals accepted into in the k rooms, then $stab\#(a_i, R) = k - i + 1$.*

The proof is similar to CLAIM 1 in the proof of Theorem 2(a). Indeed, the m-room generalization of Theorem 2(a) follows from this.

Pairs. Again we consider the sequence of state pairs in (9) corresponding to $Sort(A^+)$ and $Sort(A^0)$. Let $P = \begin{bmatrix} \boldsymbol{s} \\ \boldsymbol{t} \end{bmatrix} = \begin{bmatrix} a_1 < a_2 < \cdots < a_k \\ b_1 < b_2 < \cdots < b_\ell \end{bmatrix}$ be such a pair. Let $\mathrm{supp}(P) = \mathrm{supp}(\boldsymbol{s}) \cup \mathrm{supp}(\boldsymbol{t})$ and $rank(P) = \max\{rank(\boldsymbol{s}), rank(\boldsymbol{t})\}$. Two states $\boldsymbol{s}, \boldsymbol{t}$ are **equivalent** if there exist a monotone map $T : \mathrm{supp}(\boldsymbol{s}) \to \mathrm{supp}(\boldsymbol{t})$ such that $T(\boldsymbol{s}) = \boldsymbol{t}$. Two pairs P, Q are **equivalent** if there is a monotone map $T : \mathrm{supp}(P) \to \mathrm{supp}(Q)$ such that $T(P) = Q$. Say P is **canonical** if $\mathrm{supp}(P) = \{1, 2, \ldots, k\}$ where $k = rank(P)$. Clearly, every pair is equivalent to a unique canonical pair.

Superclasses: In order to achieve a general analysis, we cannot naively generalize Lemma 6 because we would be analyzing transition graphs with $\Omega(m^2)$ nodes. Instead, define the following set of **superclasses**:

$$V_m := \{[A_1], [B_1], [A_0], [B_0], [E]\}. \tag{21}$$

Each superclass is a sets of pairs of rank $\leqslant m$:

- $[A_1]$ is the set of all pairs $\left[\begin{smallmatrix} \boldsymbol{s} \\ \boldsymbol{t} \end{smallmatrix}\right]$ such that $(\exists a)[\mathrm{supp}(\boldsymbol{s}) = \mathrm{supp}(\boldsymbol{t}) \cup \{a\}]$.
- $[B_1]$ is the set of all pairs $\left[\begin{smallmatrix} \boldsymbol{s} \\ \boldsymbol{t} \end{smallmatrix}\right]$ such that $(\exists a < b)[\mathrm{supp}(\boldsymbol{s}) \cup \{b\} = \mathrm{supp}(\boldsymbol{t}) \cup \{a\}]$.
- $[A_0]$ is the set of all pairs $\left[\begin{smallmatrix} \boldsymbol{s} \\ \boldsymbol{t} \end{smallmatrix}\right]$ such that $(\exists b)[\mathrm{supp}(\boldsymbol{s}) \cup \{b\} = [\mathrm{supp}(\boldsymbol{t})]$.
- $[B_0]$ is the set of all pairs $\left[\begin{smallmatrix} \boldsymbol{s} \\ \boldsymbol{t} \end{smallmatrix}\right]$ such that $(\exists a > b)[\mathrm{supp}(\boldsymbol{s}) \cup \{b\} = \mathrm{supp}(\boldsymbol{t}) \cup \{a\}]$.
- $[E]$ is set of all equality pairs.

The superclasses are no longer equivalence classes. E.g., $\left[\begin{smallmatrix} 2<3<4 \\ 1<3<4 \end{smallmatrix}\right] \neq \left[\begin{smallmatrix} 1<3<4 \\ 1<2<4 \end{smallmatrix}\right]$ but both pairs belong to $[B_1]$.

▸ **Observation 8.**

(a) *Each superclass $\alpha \in V_m$ is a union of equivalence classes of pairs.*

(b) *The critical pair P_{i^+} in (9) belongs to either $[A_1]$ or $[B_1]$.*

The transition graph $G_m = (V_m, E_m)$: We have an edge $(\alpha \to \beta) \in E_m$ iff there exist pairs $P \in \alpha, Q \in \beta$ and interval I such that $P \xrightarrow{I} Q$. This graph is shown in Figure 2. Furthermore, if $P \xrightarrow[xy]{I} Q$ (for some $xy \in \{a, e, r\}^2 = \{acc, ext, rej\}^2$) then we write $\alpha \xrightarrow[xy]{} \beta$ and say[5] xy is **applicable** to α. For example, the type rr is always applicable to any superclass. We call rr the **trivial type** because always it represents self-loops: $\alpha \xrightarrow[rr]{} \alpha$. The following theorem enumerates all non-trivial types applicable to each $\alpha \in V_m$.

[5] But β may not be uniquely determined by α and xy. E.g., Theorem 9(II) shows that $[B_1] \xrightarrow[aa]{} [E]or[B_1]$.

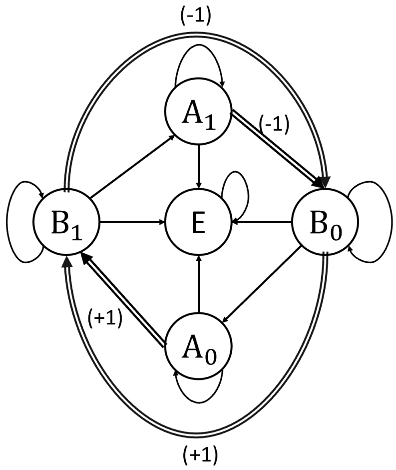

Figure 2 Transition Graph G_m

▸ **Theorem 9.**
The complete list of non-trivial types applicable to each $\alpha \in V_n$ is enumerated as follows:

(I) $[A_1] \xrightarrow[aa/ae/ee/re]{} [A_1]/[E]/[A_1]/[B_0]$

(II) $[B_1] \xrightarrow[aa/ra/ee/ea]{} [E] or [B_1]/[B_0]/[B_1]/[A_1]$

(III) $[A_0] \xrightarrow[aa/ea/ee/er]{} [A_0]/[E]/[A_0]/[B_1]$

(IV) $[B_0] \xrightarrow[aa/ar/ee/ae]{} [B_0] or [E]/[B_1]/[B_0]/[A_0]$

(V) $[E] \xrightarrow[aa/ee]{} [E]/[E]$

Proof. See appendix. **Q.E.D.**

We summarize our main result as follows:

▸ **Theorem 10** (Main Result). *The multiroom activity selection problem can be optimally solved by our greedy algorithm as defined by (20). This algorithm can be implemented in $O(n \log n)$.*

Proof. The correctness of the greedy algorithm follows from the above generalization of Theorem 2. The main issue is the complexity claim. If m is fixed, this is no issue. But in the generalized problem, m is part of the input ($m \leqslant n$). Above, we said that we maintain the invariant (19) using an array $front[1..m]$. Then it $O(m)$ to update this array for each interval I_i, or a total of $\Omega(mn)$ time. This is not acceptable if $m = \Omega(\log n)$. Our solution is fairly standard: suppose only k rooms are currently in use ($k \leqslant m$). Use any balanced binary tree T (e.g., an AVL tree) to store the pairs $\{(fTime[1], 1), \ldots, (fTime[k], k)\}$, sorted by the first component $fTime[i]$ of each pair. Given interval I, we use T to find the node $(fTime[j], j)$ ($j = 1, \ldots, k$) such that $fTime[j]$ is the largest value satisfying $fTime[j] \leqslant s(I)$. If such a j exists, we can delete node containing $(fTime[j], j)$ and insert the pair $(f(I), j)$. We can update $fTime[j] \leftarrow f(I)$. Suppose j does not exists: there are two possibilities: if $k = m$, we will reject I and there is nothing to do. Otherwise, we extend the current state by inserting the pair $(f(I), k + 1)$ into T, update $fTime[k + 1] \leftarrow f(I)$. Thus we can implement the transformation (20) in $O(\log m) = O(\log n)$ time. **Q.E.D.**

4 Conclusion

This paper introduced a novel generalization of the activity selection problem, and gave a relatively simple optimal greedy algorithm with an interesting proof. Our multi-room scenario opens up many possibilities for generalization. Which of these generalizations remains subquadratic in complexity? For instance, suppose the m rooms come in 2 sizes (big and small) and some activities can only be assigned to big rooms. Is there still an optimal greedy algorithm?

A Appendix: Full Proof of Theorem 9 and References

Theorem 9

The complete list of non-trivial types applicable to each $\alpha \in V_n$ is enumerated as follows:

(I) $[A_1] \xrightarrow[aa/ae/ee/re]{} [A_1]/[E]/[A_1]/[B_0]$

(II) $[B_1] \xrightarrow[aa/ra/ee/ea]{} [E]\text{or}[B_1]/[B_0]/[B_1]/[A_1]$

(III) $[A_0] \xrightarrow[aa/ea/ee/er]{} [A_0]/[E]/[A_0]/[B_1]$

(IV) $[B_0] \xrightarrow[aa/ar/ee/ae]{} [B_0]\text{or}[E]/[B_1]/[B_0]/[A_0]$

(V) $[E] \xrightarrow[aa/ee]{} [E]/[E]$

Proof. Case (V) is immediate. Due to the symmetry between A_1 and A_0, and between B_1 and B_0, we only need to prove Cases (I) and (II). Cases (III) and (IV) may be derived by 3 simultaneous exchanges $A_1 \leftrightarrow A_0$, $B_1 \leftrightarrow B_0$, and $xy \leftrightarrow yx$. to any transition $\alpha \xrightarrow[xy]{} \beta$.

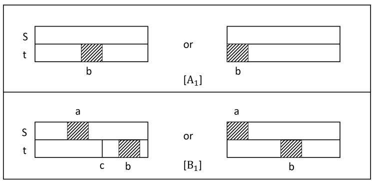

Figure 3 Canonical pairs for $[A_1]$ and $[B_1]$: two variants of each are shown.

Case (I) Refer to Figure 3(top) illustrating a canonical pair $P = \left[\begin{smallmatrix} \boldsymbol{s} \\ \boldsymbol{t} \end{smallmatrix}\right] \in [A_1]$. Let $\boldsymbol{s} = (a_1 < \cdots < a_k)$, $\boldsymbol{t} = (b_1 < \cdots < b_{k-1})$ and $b = \text{supp}(\boldsymbol{s}) \backslash \text{supp}(\boldsymbol{t})$. The figure distinguishes two possibilities: $b = a_1$ or $b > a_1$. Consider the transition $P \xrightarrow{I} Q$ for some applicable interval I.

We consider 3 cases, depending on whether $\boldsymbol{s}$ accepts, extends to accept or rejects I:

(I.1) If this interval is accepted by state $\boldsymbol{s}$, then $\boldsymbol{t}$ cannot reject I. So $\boldsymbol{t}$ can accept I, or it can extend to accept I. (i) If $\boldsymbol{t}$ accept I, then we see that Q belongs to $[A_1]$. (ii) If $\boldsymbol{t}$ extends to accept I, this implies $b_1 > b = a_1$, and $a_1 < s(I) < b_1$. Moreover, $a_1 \in \boldsymbol{s}$ is displaced by $f(I)$, and $\boldsymbol{t}$ adds $f(I)$. If $Q = \left[\begin{smallmatrix} \boldsymbol{s}' \\ \boldsymbol{t}' \end{smallmatrix}\right]$, this means that $\boldsymbol{s}' = \boldsymbol{t}'$ i.e. Q belongs to $[E]$. Thus we have shown

$$[A_1] \xrightarrow[aa/ae]{} [A_1]/[E].$$

(I.2) If $\boldsymbol{s}$ extends to accept the interval, then $\boldsymbol{t}$ must extend to accept I. So both $\boldsymbol{s}$ and $\boldsymbol{t}$ simply add the element $f(I)$. Then Q belongs to $[A_1]$. This proves that

$$[A_1] \xrightarrow[ee]{} [A_1].$$

(I.3) If $\boldsymbol{s}$ rejects I, then $\boldsymbol{t}$ must extend to accept I. After adding $f(I)$ to $\boldsymbol{t}$, the resulting pair satisfies $\text{supp}(\boldsymbol{s}) \cup f(I) = \text{supp}(\boldsymbol{t}) \cup b$. Since $b < f(I)$, Q belongs to $[B_0]$. This proves that

$$[A_1] \xrightarrow[re]{} [B_0].$$

Case (II) Refer to Figure 3(bottom) illustrating a canonical pair $P = \left[\begin{smallmatrix} \boldsymbol{s} \\ \boldsymbol{t} \end{smallmatrix}\right] \in [B_1]$. By definition of $[B_1]$, we may assume that $\boldsymbol{s} = (a_1 < \cdots < a_k)$, $\boldsymbol{t} = (b_1 < \cdots < b_k)$ and $\mathrm{supp}(\boldsymbol{s}) \cup \{b\} = \mathrm{supp}(\boldsymbol{t}) \cup \{a\}$ for some $a < b$. Also, there is a unique $c \in \mathrm{supp}(\boldsymbol{t})$ that precedes b. The figure distinguishes two possibilities: $a = b_1$ or $a > b_1$. Assume I is applicable to P, i.e., $f(I) > a_k$. Consider 3 possibilities: $\boldsymbol{s}$ accepts I, or rejects I, or extends to accepts I.

(II.1) If I is accepted by $\boldsymbol{s}$, then it must also be accepted by $\boldsymbol{t}$. But, what is the superclass of Q where $P \xrightarrow[aa]{I} Q$? This turns out to be non-unique.
Let $d \in \mathrm{supp}(\boldsymbol{s})$ be displaced when $\boldsymbol{s}$ accepts I. Similarly, let $d' \in \mathrm{supp}(\boldsymbol{t})$ be displaced when $\boldsymbol{t}$ accepts I. If $d \neq b$ then $d' = d$, and we see that Q still belongs to $[B_1]$. If $d = b$, then we see that $d' = c$. Now, there are two possibilities: either $a < c$ in which case $Q \in [B_1]$, or $a = c$ in which case $Q \in [E]$. This proves our claim that

$$[B_1] \xrightarrow[aa]{} [E] \mathrm{or} [B_1].$$

(II.2) If $\boldsymbol{s}$ rejects I, then $\boldsymbol{t}$ must also reject I unless $a = b_1$. If $a = b_1$, $\boldsymbol{t}$ may accept I but it may not extend $\boldsymbol{t}$ to accept I. Accepting I implies that b_1 is displaced by $f(I)$ in Q. Then Q belongs to $[B_0]$. This proves that

$$[B_1] \xrightarrow[ra]{} [B_0].$$

(II.3) If $\boldsymbol{s}$ extends to accept I, then $\boldsymbol{t}$ cannot reject I. So $\boldsymbol{t}$ can also extend to accept I, or it can accept I: (i) If $\boldsymbol{t}$ extends to accept I, then we see that Q belongs to $[B_1]$. (ii) If $\boldsymbol{t}$ accepts I, this implies $b_1 = a$. Moreover, b_1 is displaced by $f(I)$ in $\boldsymbol{t}$. If $Q = \left[\begin{smallmatrix} \boldsymbol{s}' \\ \boldsymbol{t}' \end{smallmatrix}\right]$, this means that $\mathrm{supp}(\boldsymbol{t}') = \mathrm{supp}(\boldsymbol{t}') \cup \{b\}$, i.e., $Q \in [A_1]$. Thus we have shown

$$[B_1] \xrightarrow[ee/ea]{} [B_1]/[A_1].$$

Q.E.D.

References

1 Thomas H. Corman, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press and McGraw-Hill Book Company, Cambridge, Massachusetts and New York, fourth edition, 2022.

2 Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, revised 3rd edition edition, 2008.

3 F. Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques and maximum independent set of a chordal graph. *SIAM J. Computing*, 1:180–187, 1972.

4 Jon Kleinberg and Éva Tardos. *Algorithm Design*. Addison Wesley, Boston, 2006.

5 Kurt Mehlhorn and Peter Sanders. *Algorithms and Data Structures: The Basic Toolbox*. Springer, Berlin-Heidelberg, 2008.

6 D. Rose, George Lueker, and Robert E. Tarjan. Algorithmic aspects of vertex elimination on graphs. *SIAM J. Computing*, 5(2):266–283, 1976. doi:doi:10.1137/0205021.