1 Maximum Matching - greedy algorithm

Theorem 1.1. The maximum matching greedy algorithm for interval graphs is correct.

Proof. Let $I := \{i_1, ...i_n\}$; with the intervals sorted by their right-end point. The proof is done by induction on $|I|$.
For $|I| = 1$, the algorithm gives $M = \emptyset$; which is correct.

Induction step

Suppose the algorithm gives the correct answer for $n-1$ intervals. Let $|I| = n$, and M be the matching computed by the greedy algorithm for $I - i_n$. Let U be the set of unmatched intervals.
If U is empty, when treating I the algorithm will give the same answer: M. Let U be non empty. Let $i_k \in U$. If $i_k \cap i_n \neq \emptyset$, the algorithm will return $M + (i_k, i_n)$. Indeed, for $i < k$:

- If i_i could be matched with something in $I - i_n$, it would never choose to match it with i_n as it is the worst choice.
- If i_i could not be matched in $I - i_n$, then i_i will be left unmatched; because if it was possible to match it with i_n; then i_i and i_k would have not be unmatched $I - i_n$.

Augmentating path

Suppose now that U is non empty, and that for some $i_k \in U$ we can find a shortest augmentating path $P : (i_k, j_1, j'_1, j_2, j'_2, ..., j'_m, i_n)$, namely:

- $i_k \cap j_1 \neq \emptyset$ and $j'_m \cap i_n \neq \emptyset$.
- $\forall x \in [1, m-1], j'_x \cap j_{x+1} \neq \emptyset$.
- $\forall x \in [1, m-1], (j_x, j'_x) \in M$.

Until the end of the proof, we will use the notation r_k, r_x, r'_x for the right-end points of respectively i_k, j_x, j'_x.

Decreasing property

We shall now prove by a second induction that $\forall x \in [0, m-1], r'_{x+1} < r'_x$ and $r_{x+1} < r_x$. We only need the first property to conclude, but we need both to do the proof.
First let's prove this for j_1, j'_1, j_2 and j'_2:

Observe that if both j_1, j'_1 overlap i_k, then: $r_1 < r_k$ and $r'_1 < r_k$; otherwise, i_k would have matched with either j_1 or j'_1. It is also possible that $r_1 > i_k$; and in that case $r'_1 < r_k$ and $j'_1 \cap i_k = \emptyset$.
Suppose for a moment that $r_2 > r_1$:

- case 1: if $r'_1 < r_1$, then $j_2 \cap i_k \neq \emptyset$, so we can cut j_1, j'_1 from P.
- case 2: if $r_1 < r'_1$, then either j_2 overlap i_k as in case 1; either j_1 and i_k would be matched as in Fig 2.
- case 3: if $r_1 > r_k$, then $j_2 \cap i_k \neq \emptyset$, so we can cut j_1, j'_1 from P.

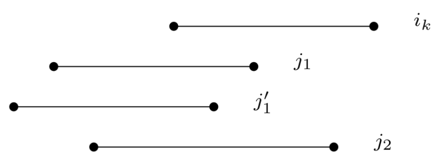


Figure 1: Skipping j_1, j'_1 would be shorter.

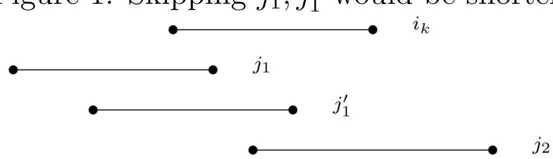
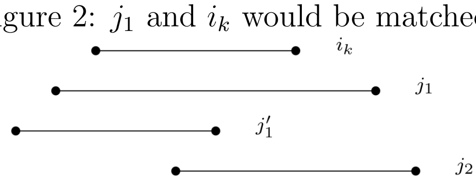


Figure 2: j_1 and i_k would be matched.



Now we know that $r_2 < r_1$. Let's suppose for a moment that $r'_2 > r'_1$. Notice that the smallest element among $\{r_1, r_2, r'_1, r'_2\}$ is either r'_1 or r_2. In any case the algorithm would have match the intervals differently.

Induction step

Suppose $x > 1$, $r_x < r_{x+1}$ and $r'_x < r'_{x+1}$.
We will investigate four different cases:

- 1.1 : $r_{x-1} > r'_{x-1}$ and $r_x > r'_x$
- 1.2 : $r_{x-1} > r'_{x-1}$ and $r'_x > r_x$
- 2.1 : $r'_{x-1} > r_{x-1}$ and $r_x > r'_x$
- 2.2 : $r'_{x-1} > r_{x-1}$ and $r'_x > r_x$

Case 1.1 One of two things are possible:

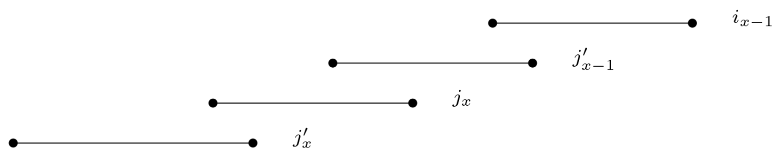


Figure 3: Subcase 1.1.1: $r_x < r'_{x-1}$

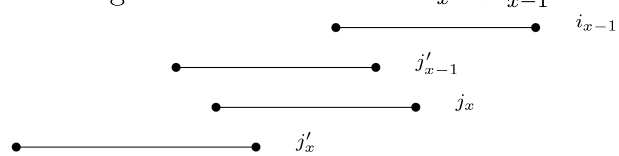


Figure 4: Subcase 1.1.2: $r_x > r'_{x-1}$

For Subcase 1.1.1, as $r_x < r'_{x-1}$, it means that $r_x \in j'_{x-1}$. Consequently, if $r_{x+1} > r'_x$, we must have $r_x \in j_{x+1}$. In other words, j'_{x-1} and j_{x+1} share a common point in r_x, so we could remove j_x, j'_x from P, so we must have $r_{x+1} < r_x$.
Suppose now that $r'_{x+1} > r'_x$: observe that as $r_{x+1} < r_x$, the algorithm has an incentive to match j'_x with j_{x+1}; and the only reason they are not matched is because j_{x+1} and j'_{x+1} were matched first in the execution, ie $r_{x+1} < r'_{x+1}$.

For subcase 1.1.2: suppose for a moment that $r_{x+1} > r_x$. The exact same argument with r'_{x-1} in place of r_x as in 1.1.1 still holds here, so $r_{x+1} < r_x$.
Suppose now that $r'_{x+1} > r'_x$. Then the algorithm should have matched j_{x+1} and j'_x. Thus $r'_{x+1} < r'_x$ for case 1.1.

Case 1.2

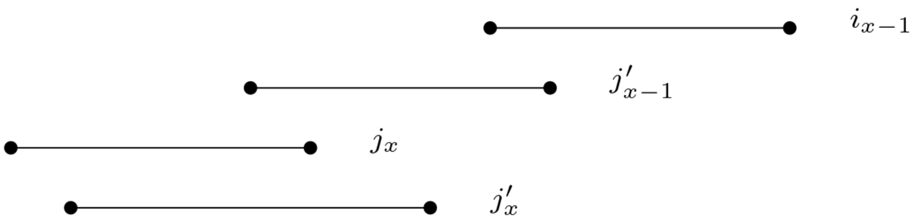


Figure 5: $r_{x-1} > r'_{x-1}$ and $r'_x > r_x$

Let l_{x+1} be the left end-point of j_{x+1}. Suppose $r_{x+1} > r_x$. If $l_{x+1} < r_x$, then $r_x \in j_{x+1}$. As $r_x \in j'_{x-1}$, it means we could remove j_x, j'_x from P.
If $l_{x+1} > r_x$, notice we should at least have $l_{x+1} < r'_x$. As $r'_x < r'_{x-1}$, we have $r'_x \in j'_{x-1}$. So $l_{x+1} \in [r_x, r'_x] \subset j'_{x-1}$; meaning we could remove j_x, j'_x from P.
Thus, $r_{x+1} < r_x$.
As $r'_{x+1} < r'_x$, the algorithm has an incentive to match j'_x with j_{x+1}. As for the 1.1.x cases, the only reason for it to not happen is that j_{x+1} is already matched with j'_{x+1}, because $r'_{x+1} < r'_x$.

Case 2.1

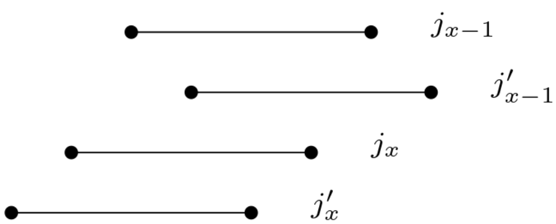


Figure 6: $r'_{x-1} > r_{x-1}$ and $r_x > r'_x$

If $r_{x+1} > r_x$, then we can skip j_x, j'_x as in case 1.1. So we must have $r_{x+1} < r_x$.
If $r'_{x+1} > r'_x$, then j_{x+1} would be matched with j'_x as it ends before j_{x+1}. Thus $r'_{x+1} < r'_x$.

Case 2.2

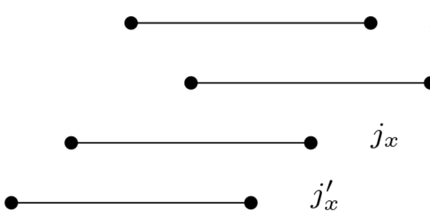


Figure 7: $r'_{x-1} > r_{x-1}$ and $r'_x > r_x$

Suppose $r_{x+1} > r_x$. Using similar arguments as in case 1.2, we should conclude that we can remove j_x, j'_x from P. So $r_{x+1} < r_x$.
Suppose now $r'_{x+1} > r'_x$. Then j_{x+1} would be matched with j'_x (or j_x if these overlaps).

Conclusion
We have showned by induction that $\forall x > 1, r'_{x+1} < r'_x$. This is a contradiction with the properties of P: there is no way j'_m could possibly overlap i_n. There is no augmentating path; thus, the greedy algorithm is optimal.

$\square$