Fast A* Algorithms for Multiple Sequence Alignment

Takahiro Ikeda
ike@is.s.u-tokyo.ac.jp

Hiroshi Imai
imai@is.s.u-tokyo.ac.jp

Department of Information Science, Faculty of Science,
University of Tokyo
7–3–1 Hongo, Bunkyo-ku, Tokyo 113, Japan

Abstract

The multiple alignment of the sequences of DNA and proteins is applicable to various important fields in molecular biology. Although the approach based on Dynamic Programming is well-known for this problem, it requires enormous time and space to obtain the optimal alignment. On the other hand, this problem corresponds to the shortest path problem and the A* algorithm, which can efficiently find the shortest path with an estimator, is usable.

This paper directly applies the A* algorithm to multiple sequence alignment problem with more powerful estimator in more than two dimensional case and discusses the improvement of this approach utilizing an upper bound of the shortest path length. The algorithm to provide the upper bound is also proposed in this paper.

1 Introduction

The multiple sequence alignment is the problem to find the alignment of multiple sequences with highest score due to a given scoring criterion between characters. The solution of this problem for multiple sequences of DNA and proteins represent the similarity of them and are applicable to various important fields such as the prediction of three dimensional structures of proteins and the inference of phylogenetic tree in molecular biology.

This problem can be solved by finding the shortest path on some directed acyclic graph. Suppose that S_k denotes the k-th sequence whose length is $n_k = O(n)$ and d denotes the dimension, the number of sequences. Then in the directed acyclic graph $G = (V, E)$ such that $V = \{(x_1, \ldots, x_d) \mid x_i = 0, 1, \ldots, n_i\}$ and $E = \cup_{e \in \{0,1\}^d} \{(v, v + e) \mid v, v + e \in V,\ e \neq 0\}$, a path from the vertex $s = (0, \ldots, 0)$ to the vertex $t = (n_1, \ldots, n_d)$ corresponds to an alignment of sequences.

池田 崇博, 今井 浩: 東京大学理学部情報科学科, 〒 113 東京都文京区本郷 7–3–1

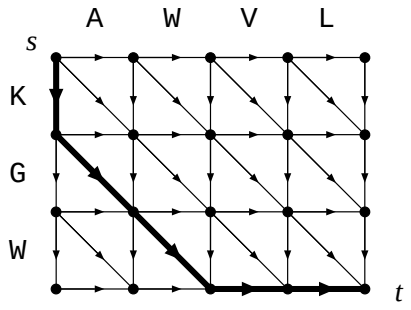

Figure 1: The graph for the alignment of two sequences, KGW and AWVL. The path from s to t drawn as the bold line represents the alignment of KGW-- and -AWVL.

For instance, in two dimensional case, the graph G is constructed as Figure 1. In this graph, each row and column correspond to each character of first and second sequences, respectively. A diagonal edge represents a match between two characters and both horizontal and vertical edges represent insertions of gaps. Therefore finding the optimal alignment is equivalent to finding the shortest path from the top left vertex to the bottom right vertex on an appropriate assignment of edge length such that matching cost of two characters, which denotes less similarity of them, is assigned to each diagonal edge and the gap cost is assigned to each horizontal and vertical edge. If matching scores representing the similarity of characters are given, matching costs are obtained by reversing their signs.

In more than two dimensional case, the sum of all scores for pairwise sequence alignments is used as the score for the multiple sequence alignment in general. This corresponds to defining each edge length in the original graph $G = (V, E)$ as the sum of all corresponding edge length in the graphs for pairwise alignments. Let $G_{ij} = (V_{ij}, E_{ij})$ denote the graph for the alignment of S_i and S_j $(i < j)$, that is, $V_{ij} = \{v_{ij} = (x_i, x_j) \mid v = (x_1, \ldots, x_d) \in V\}$ and $E_{ij} = \{(u_{ij}, v_{ij}) \mid (u, v) \in E, u_{ij} \neq v_{ij}\}$. Then the length of edge (u, v) in E is defined as $l(u, v) = \sum_{1 \leq i < j \leq d} l(u_{ij}, v_{ij})$ where $l(u_{ij}, v_{ij})$ denotes the length of edge (u_{ij}, v_{ij}) in graph G_{ij} and has been defined.

The method based on Dynamic Programming (DP) is a faithful approach for multiple sequence alignment problem. This method searches all vertices in the graph and has $O(n^d)$ time and space complexity. Although this approach is effective for small dimension of two or three, it is impractical to apply this method directly to higher dimensional problem because n^d is enormous even for a little larger d. In order to avoid this tendency, approximate methods are used for higher dimensional alignment problem since it is difficult to bound the search space without lack of optimality of the result alignment.

On the other hand, the A* algorithm, which is the well-known heuristic search method in Artificial Intelligence, always finds the optimal alignment with less vertices searched [2, 4]. It reduces unnecessary search by utilizing the heuristic estimate for the shortest path length from each vertex to the destination.

The concept of this algorithm has been used in multiple sequence alignment problem to bound the search space of DP [5]. Although the A* algorithm is incompatible with DP, this bounding method overcomes this problem by using an upper bound of the shortest path length.

Therefore, this method requires additional time to obtain an upper bound compared with the original A* algorithm and besides its search space is not less than that of the original A* algorithm.

On the other hand, the A* algorithm can be directly used in finding the shortest path on the graph for multiple sequence alignment. In two dimensional case, the A* algorithm has been directly applied and its effectiveness has been reported [1](see also [3]).

This paper focuses on more than two dimensional case, and shows the A* algorithm can find the shortest path by only searching sufficiently small space in the actual application to protein sequence alignment. This paper further improves this approach in order to decrease the necessary space using an upper bound for the shortest path length. This algorithm is never inferior to the approach based on DP using an upper bound with regard to the necessary space. This paper also proposes an efficient approximate algorithm which provides the almost optimal alignment with less necessary time and space. This algorithm is applicable to finding an upper bound as well.

2 The A* Algorithm

The A* Algorithm finds the shortest path from s to t efficiently when all edge length is non-negative. It utilizes a heuristic estimate for the shortest path length from each vertex to t, which must be at most the actual shortest path length, and reduces the search space. Let $h(v)$ denote this estimate value for a vertex v. Then the outline of the A* Algorithm is described as follows [2, 4]:

1. Let U be the set $\{s\}$ and $p(s)$ be zero.
2. Find the vertex u which has the minimum value of $p(v) + h(v)$ in U and remove u from U. If u equals to t, then halt.
3. For all vertices v such that (u, v) is in E, if $p(u)+l(u, v)$ is less than $p(v)$, replace the path from s to v with the path from s to u and the edge (u, v), and let $p(v)$ be $p(u) + l(u, v)$, and add v to U if v is not in U.
4. Go to step 2.

This algorithm is a type of the Dijkstra method using $p(v) + h(v)$ instead of $p(v)$ as the criterion of the search order for vertex v. The potential $p(v)$ represents the temporary shortest path length from s to v and the expression $p(v) + h(v)$ represents the temporary estimate for the shortest path length from s to t via v. Hence it searches vertices near the shortest path preferentially and reduces the number of vertices to be searched with an appropriate estimator.

In this algorithm, set U always keeps candidates for the vertex to be selected next and may be compared to the wave front of the search. This set is managed with the operations in steps 2 and 3. This series of operations is called an expansion of the vertex.

In the A* algorithm, the shortest path from s is not always fixed for the expanded vertex unlike the Dijkstra method, that is, shorter paths may be found in future search for such vertices. This induces the operation to return the expanded vertex to U in step 3 and increases the expansions of vertices in the A* algorithm. However, this disadvantage of the A* algorithm

is due to the feature of the estimator and can be eliminated if the estimator is dual feasible as the following definition.

Definition 1 *The estimator h is dual feasible if and only if h satisfies the following condition for each edge (u, v) in E:*

$$l(u, v) + h(v) \geq h(u).$$

If the estimator is dual feasible, the A* algorithm is the same as the Dijkstra method except for the criterion of the search order. In fact, the A* algorithm can be transformed into the Dijkstra method by modifying edge length as the following proposition.

Proposition 1 *The Dijkstra method with the edge length l' modified as follows is equivalent to the A* algorithm with the original edge length l utilizing a dual feasible estimator h:*

$$l'(u, v) = l(u, v) + h(v) - h(u).$$

Proof: Since $l'(u, v)$ is non-negative from dual feasibility of h, it is possible to use $l'(u, v)$ as the length of an edge (u, v) in the Dijkstra method. Let $P(s, v)$ be the temporary path from s to a vertex v. Then the following formula is satisfied on the potential of arbitrary expanded vertex v in the Dijkstra method:

$$p(v) = \sum_{(x,y)\in P(s,v)} l'(x, y) = \sum_{(x,y)\in P(s,v)} l(x, y) + h(v) - h(s).$$

Notice that $h(s)$ is a constant and $\sum_{(x,y)\in P(s,v)} l(x, y)$ denotes the potential of v in the A* algorithm. This indicates the Dijkstra method using modified edge length is equivalent to the A* algorithm. □

In this modification, each new edge length is always positive only if the estimator is dual feasible. This means the Dijkstra method transformed from A* algorithm can work correctly with a dual feasible estimator even if the graph has negative length edges.

3 The A* Algorithm for Multiple Sequence Alignment

3.1 An Indirect Application of the A* Algorithm

As the application of the A* algorithm to multiple sequence alignment problem, Spouge has proposed the approach to bound the search space using the feature of the A* algorithm [5]. This approach is based on the idea to apply DP to only vertices to be searched by the A* algorithm and can be regarded as the indirect application of the A* algorithm. This section reviews this approach and discusses its merits and demerits.

Let $L^*(u, v)$ denote the shortest path length from u to v and $L^u(u, v)$ denote its upper bound. Spouge has shown that searching only such vertices v that $p(v) + h(v) \leq L^u(s, t)$ is sufficient for finding the shortest path from s to t. This is due to the fact that the A* algorithm expands the vertex v in order of the value $p(v) + h(v)$ which is the estimate for the shortest path length from s to t via v and does not exceed the actual shortest path length $L^*(s, t)$. This means any vertex v expanded by the A* algorithm satisfies $p(v)+h(v) \leq L^*(s, t) \leq L^u(s, t)$ and

guarantees the shortest path from s to t only passes such vertex v that $p(v) + h(v) \leq L^u(s,t)$. Then applying DP to such vertices completes the algorithm. This paper calls this algorithm as enhanced DP in convenience.

This approach can be regarded as coercive implementation of the A* algorithm with DP. The merit of this approach is that it can omit the management of set U in the A* algorithm. The A* algorithm requires set U in order to maintain the search order and uses the operations to extract the element with minimum key and to decrease the key of specified element for set U. These operations take $O(\log |U|)$ time even if a binary heap is utilized as the implementation of U. On the other hand, DP searches vertices in order specified in advance and does not require additional data such as U.

The demerit of enhanced DP is that it uses an upper bound of the shortest path length in bounding its search space. Although the effect of this bounding is due to tightness of the upper bound, it is expensive to obtain a tighter upper bound. If it takes a lot of time to calculate an upper bound, the merit of enhanced DP will be canceled out. In addition to this, the number of searched vertices in enhanced DP is larger than the number of expanded vertices in the A* algorithm unless the upper bound is tight. These numbers correspond to the number of iterations in each algorithm, and are proportional to the execution time of each algorithm. As the upper bound becomes looser, the difference between them becomes larger. Although Spouge has proposed to modify the upper bound $L^u(s,t)$ into $L^*(s,v) + L^u(v,t)$ dynamically at some vertex v in order to make the upper bound tighter, it requires still more execution time.

These demerits of enhanced DP are due to the incompatibility between DP and the A* algorithm. Although the essence of the A* algorithm is in its search order, it is broken in the search with enhanced DP. In this way, enhanced DP loses the feature of the A* algorithm although it gains the feature of DP the A* algorithm does not have.

3.2 A Direct Application of the A* Algorithm

In consideration of the discussion in the previous section, this paper takes an approach to apply the A* algorithm directly to the graph for multiple sequence alignment. This approach can omit the calculation of the upper bound and keep the expanded vertices less although it requires additional time and space to maintain the candidates for the vertex to be expanded next. This section discusses this approach with attention to the estimator of the A* algorithm.

For two dimensional problem, the A* algorithm has been directly applied [1]. In these cases, the estimators are constructed from information on the alignment of characters. Although this construction is reasonable and appropriate for two dimensional problem, more powerful estimator can be constructed for more than two dimensional problem. The efficiency of the A* algorithm strongly depends on the performance of the estimator. More exact estimator reduces unnecessary search and enables the A* algorithm to find the shortest path faster.

This paper focuses on more than two dimensional problem and adopts the same estimator has been proposed for enhanced DP by Spouge [5]. The following h denotes this estimator:

$$h(v) = \sum_{1 \leq i < j \leq d} L^*(v_{ij}, t_{ij}).$$

Recall that the length of a path in a multiple alignment problem is defined as the sum of all length of corresponding paths in pairwise alignment problems. This estimator uses the shortest

path length in the pairwise alignment problem as the estimate for the length of the path corresponding to the shortest path in the multiple alignment problem. In higher dimensional problem, necessary time and space for solving pairwise problems is negligible compared with those for solving a multiple problem and does not increase time and space complexity. Therefore it is possible to use the estimator which provides more exact estimates utilizing information on all pairwise alignments.

It is clear that each estimate $h(v)$ does not exceed the actual shortest path length from v to t and this estimator h is available in the A* algorithm. Moreover, the following proposition is satisfied on dual feasibility of h.

Proposition 2 *This estimator h is dual feasible.*

Proof: For any edge (u, v) in E,

$$l(u,v) + h(v) = \sum_{1 \le i < j \le d} (l(u_{ij}, v_{ij}) + L^*(v_{ij}, t_{ij})) \ge \sum_{1 \le i < j \le d} L^*(u_{ij}, t_{ij}) = h(u).$$

This is the definition of dual feasibility itself. □

Hence the A* algorithm using this estimator is applicable to graphs which have negative length edges with the transformation based on Proposition 1. Finally, this paper proposes the following approach.

1. For arbitrary pair of i and j satisfying $1 \le i < j \le d$, apply DP to graph G_{ij} from vertex t_{ij} and calculate $L^*(v_{ij}, t_{ij})$ for any v_{ij} in V_{ij}.
2. Apply the Dijkstra method to graph G from vertex s with the length of edge (u, v) modified as $l(u, v) + h(v) - h(s)$ where $h(v) = \sum_{1 \le i < j \le d} L^*(v_{ij}, t_{ij})$.

In order to investigate the actual efficiency of this approach, the experiment aligning actual sequences of proteins has been performed. In this experiment, elongation factor TU (EF-TU) of Haloarcula marismortui (Hal) and Methanococcus vannielii (Met), and elongation factor 1α (EF-1α) of Thermoplasma acidophilum (Tha), Thermococcus celer (Thc), Sulfolobus acidocaldarius (Sul), Entamoeba histolytica (Ent) and Plasmodium falciparum (Pla) have been used as proteins and the optimal alignments of first d sequences of them have been calculated based on this approach for $3 \le d \le 7$. The PAM-250 matrix has been used in assigning edge length with each sign of score reversed. With regard to the gap penalty, the minimum value in the PAM-250 matrix, -8, has been adopted. Figure 2 illustrates the result alignment in seven dimensional case. Table 1 displays the number of vertices in the graph ($|V|$) and the number of expanded vertices (#Expanded) and the number of visited vertices (#Visited) and the maximum size of set U ($\max|U|$), and the execution time by SPARC Station 10 with 64 megabytes memory (Time). This paper regards the A* algorithm visits a vertex when the vertex is added to U. Then the number of visited vertices corresponds to the necessary space for the A* algorithm.

The result of this experiment shows this approach can find the shortest path by only expanding significantly small number of vertices while the number of vertices in the graph runs into astronomical figures. This indicates the estimator adopted in this approach provides rather exact estimate sufficiently close to the actual shortest path length. However, execution time does not reflect the number of expanded vertices.

Table 1: The result of the experiment aligning actual sequences of proteins based on the approach to apply the A* algorithm directly.

d	3	4	5	6	7
$\|V\|$	7.6×10^7	3.3×10^{10}	1.4×10^{13}	6.1×10^{15}	2.7×10^{18}
#Expanded	465	950	2,731	5,942	48,521
#Visited	3,082	9,094	32,219	104,267	838,812
max $\|U\|$	2,618	8,145	29,489	98,326	790,292
Time (*sec.*)	6	9	19	55	12,592

Table 2: The result of the experiment aligning actual sequences of proteins with enhanced A*.

d	3	4	5	6	7
#Visited	465	979	2,804	6,032	48,998
max $\|U\|$	12	374	1,621	3,784	34,898
Time (*sec.*)	6	9	15	31	319

This phenomenon is mainly due to the maximum size of U. In this experiment, a binary heap is utilized as the implementation of set U. Although this implementation is sufficiently efficient compared with an array or a list, it requires $O(\log |U|)$ time in extracting the element with minimum key and in decreasing the key of specified element with this implementation. The excessive increase of execution time is also due to the increase of the necessary space for the algorithm. This causes the shortage of real memory and leads to swapping memory. This influence is observed particularly in seven dimensional case. These two factors arise from the fact that the A* algorithm visits all adjacent vertices when it expands a vertex. This induces unnecessary addition of vertices to U and increases the necessary space blindly.

In order to weaken this undesirable tendency, this paper proposes to utilize the upper bound for the shortest path length as enhanced DP. Recall that the A* algorithm expands only such vertex v that $p(v) + h(v) \leq L^*(s,t) \leq L^u(s,t)$. This indicates the vertex violates this condition stays in U to the end if the vertex is visited by the A* algorithm. Hence it is meaningless and needless to add such vertex that $p(v) + h(v) > L^u(s,t)$ to U. This paper calls the algorithm based on this concept as enhanced A* in convenience.

The condition to bound vertices in enhanced A* is the same as that in enhanced DP. Therefore the number of visited vertices in enhanced A* never exceeds the number of searched vertices in enhanced DP and enhanced A* is no less than enhanced DP with regard to the necessary space.

Table 2 displays the result of the experiment applying enhanced A* to the same sequences of proteins in the previous experiment. In this experiment, the actual shortest path length has been utilized as the upper bound in order to examine the best possible case. Each execution time does not contain time to obtain the upper bound.

The result shows this approach sufficiently prevents the A* algorithm from adding unnecessary vertices to U. The number of visited vertices is close to the number of expanded vertices

```
     1                                                                  80
Hal  MS-DEQHQNLAIIGHVDHGKSTLVGRLLYETGSVPEHVIEQHKEEAEEKGKGGFEFAYVMDNLAEERERGVTIDIAHQEF
Met  MAKTKPILNVAFIGHVDAGKSTTVGRLLLDGGAIDPQLIVRLRKEAEEKGKAGFEFAYVMDGLKEERERGVTIDVAHKKF
Tha  MASQKPHLNLITIGHVDHGKSTLVGRLLYEHGEIPAHIIEEYRKEAEQKGKATFEFAWVMDRFKEERERGVTIDLAHRKF
Thc  MAKEKPHINIVFIGHVDHGKSTTIGRLLFDTANIPENIIKKFE-EMGEKGK-SFKFAWVMDRLKEERERGITIDVAHTKF
Sul  MS-QKPHLNLIVIGHVDHGKSTLIGRLLMDRGFIDEKTVKEAEEAAKKLGKDSEKYAFLMDRLKEERERGVTINLSFMRF
Ent  MPKEKTHINIVVIGHVDSGKSTTTGHLIYKCGGIDQRTIEKFEKESAEMGKGSFKYAWVLDNLKAERERGITIDISLWKF
Pla  MGKEKTHINLVVIGHVDSGKSTTTGHIIYKLGGIDRRTIEKFEKESAEMGKGSFKYAWVLDKLKAERERGITIDIALWKF

     81                                                                 160
Hal  STDTYDFTIVDCPGHRDFVKNMITGASQADNAVLVVAA-D---D-GV-QP-QTQEHVFLARTLGIGELIVAVNKMDLVD-
Met  PTAKYEVTIVDCPGHRDFIKNMITGASQADAAVLVVNVDDA--KSGI-QP-QTREHVFLIRTLGVRQLAVAVNKMDTVN-
Tha  ETDKYYFTLIDAPGHRDFVKNMITGTSQADAAILVISARDG--E-GV-ME-QTREHAFLARTLGVPQMVVAINKMDATSP
Thc  ETPHRYITIIDAPGHRDFVKNMITGASQADAAVLVVAV-T---D-GV-MP-QTKEHAFLARTLGINNILVAVNKMDMVN-
Sul  ETRKYFFTVIDAPGHRDFVKNMITGASQADAAILVVSAKKGEYEAGMSAEGQTREHIILSKTMGINQVIVAINKMDLADT
Ent  ETSKYYFTIIDAPGHRDFIKNMITGTSQADVAILIVAAGTGEFEAGISKNGQTREHILLSYTLGVKQMIVGVNKMDAIQ-
Pla  ETPRYFFTVIDAPGHKDFIKNMITGTSQADVALLVVPADVGGFDGAFSKEGQTKEHVLLAFTLGVKQIVVGVNKMDTVK-

     161                                                                240
Hal  -YGESEYKQVVEEV-KDLLTQVRFDSENAKFIPVSAFEGDNIAEESEHTGWYDGEILLEALNELPAPEPPTDAPLRLPIQ
Met  -FSEADYNELKKMIGDQLLKMIGFNPEQINFVPVASLHGDNVFKKSERNPWYKGPTIAEVIDGFQPPEKPTNLPLRLPIQ
Tha  PYSEKRYNEVKADA-EKLLRSIGFK-D-ISFVPISGYKGDNVTKPSPNMPWYKGPTLLQALDAFKVPEKPINKPLRIPVE
Thc  -YDEKKFKAVAEQV-KKLLMMLGYK-N-FPIIPISAWEGDNVVKKSDKMPWYNGPTLIEALDQMPEPPKPTDKPLRIPIQ
Sul  PYDEKRFKEIVDTV-SKFMKSFGFDMNKVKFVPVVAPDGDNVTHKSTKMPWYNGPTLEELLDQLEIPPKPVDKPLRIPIQ
Ent  -YKQERYEEIKKEI-SAFLKKTGYNPDKIPFVPISGFQGDNMIEPSTNMPWYKGPTLIGALDSVTPPERPVDKPLRLPLQ
Pla  -YSEDRYEEIKKEV-KDYLKKVGYQADKVDFIPISGFEGDNLIEKSDKTPWYKGRTLIEALDTMQPPKRPYDKPLRIPLQ

     241                                                                320
Hal  DVYTISGIGTVPVGRVETGILNTGDNVSFQPSD-V----S-GEVKTVEMHHEEVPKAEPGDNVGFNVRGVGKDDIRRGDV
Met  DVYTITGVGTVPVGRVETGIIKPGDKVVFEPAG-A----I-GEIKTVEMHHEQLPSAEPGDNIGFNVRGVGKKDIKRGDV
Tha  DVYSITGIGTVPVGRVETGVLKPGDKVIFLPAD-K----Q-GDVKSIEMHHEPLQQAEPGDNIGFNVRGIAKNDIKRGDV
Thc  DVYSIKGVGTVPVGRVETGVLRVGDVVIFEPASTIFHKPIQGEVKSIEMHHEPMQEALPGDNIGFNVRGVGKNDIKRGDV
Sul  EVYSISGVGVVPVGRIESGVLKVGDKIVFMPVG-K----I-GEVRSIETHHTKIDKAEPGDNIGFNVRGVEKKDVKRGDV
Ent  DVYKISGIGTVPVGRVETGILKPGTIVQFAPSG-V----S-SECKSIEMHHTALAQAIPGDNVGFNVRNLTVKDIKRGNV
Pla  GVYKIGGIGTVPVGRVETGILKAGMVLNFAPSA-V----V-SECKSVEMHKEVLEEARPGDNIGFNVKNVSVKEIKRGYV

     321                                                                400
Hal  CGPADDPPSVA--ET-FQAQIVVMQHPSVITEGYTPVFHAHTAQVACTVESIDKKIDPSSGEVAE-ENPDFIQNGDAAVV
Met  LGHTTNPPTVA--TD-FTAQIVVLQHPSVLTDGYTPVFHTHTAQIACTFAEIQKKLNPATGEVLE-ENPDFLKAGDAAIV
Tha  CGHLDTPPTVV--KA-FTAQIIVLNHPSVIAPGYKPVFHVHTAQVACRIDEIVKTLNPKDGTTLK-EKPDFIKNGDVAIV
Thc  AGHTNNPPTVVRPKDTFKAQIIVLNHPTAITVGYTPVLHAHTLQVAVRFEQLLAKLDPRTGNIVE-ENPQFIKTGDSAIV
Sul  AGSVQNPPTVA--DE-FTAQVIVIWHPTAVGVGYTPVLHVHTASIACRVSEITSRIDPKTGKEAE-KNPQFIKAGDSAIV
Ent  ASDAKNQPAVG-CED-FTAQVIVMNHPGQIRKGYTPVLDCHTSHIACKFEELLSKIDRRTGKSMEGGEPEYIKNGDSALV
Pla  ASDTKNEPAKG-CSK-FTAQVIILNHPGEIKNGYTPLLDCHTSHISCKFLNIDSKIDKRSGKVVE-ENPKAIKSGDSALV

     401                                                  455
Hal  TVRPQKPLSIEPSSEIPELGSFAIRDMGQTIAAGKV-------LG-VN-E----R
Met  KLIPTKPMVIESVKEIPQLGRFAIRDMGMTVAAGMA-------IQ-VTAK--N-K
Tha  KVIPDKPLVIEKVSEIPQLGRFAVLDMGQTVAAGQC-------ID-LE-K----R
Thc  VLRPTKPMVIEPVKEIPQMGRFAIRDMGQTVAAGMV-------IS-IQ-K--A-E
Sul  KFKPIKELVAEKFREFPALGRFAMRDMGKTVGVGVI-------ID-VKPRKVEVK
Ent  KIVPTKPLCVEEFAKFPPLGRFAVRDMKQTVAVGVV-------KA-VT------P
Pla  SLEPKKPMVVETFTEYPPLGRFAIRDMRQTIAVGIINQLKRKNLGAVTAKAPAKK
```

Figure 2: The optimal alignment of EF-TU and EF-1α calculated with the A* algorithm.

Table 3: The result of the experiment aligning seven sequences of proteins with A algorithm.

k	1	1.001	1.005	1.01	1.05	1.1
Score	24,912	24,912	24,903	24,880	24,692	24,060
#Expanded	48,521	965	617	492	456	456
#Visited	838,812	77,384	64,364	58,495	55,935	55,686
$\max \lvert U \rvert$	790,292	76,420	63,748	58,004	55,480	55,231
Time (*sec.*)	12,592	40	34	32	31	31

in all cases. The efficiency of the A* algorithm is significantly improved especially for higher dimensional case.

Although there remains the problem how to obtain a tighter upper bound in sufficiently small time, the algorithm proposed in the next section provides the solution for this problem.

4 An Approximate Method Based on the A Algorithm

The A* algorithm with no condition that each estimate must be at most the actual shortest path length is called as the A algorithm. Although the A algorithm cannot necessarily find the shortest path, it visits less vertices than the A* algorithm with an appropriate estimator.

This paper proposes to use the A algorithm with the following estimator h in calculating an upper bound for the shortest path where k is some constant:

$$h(v) = k \sum_{1 \le i < j \le d} L^*(v_{ij}, t_{ij}).$$

This estimator is k times as large as the estimator adopted in the A* algorithm. Then this algorithm regards the cost from a vertex to t k times more important than the cost from s to the vertex and searches vertices near t preferentially. Although this concept is not applicable to the graph with negative length edges, such a graph is adaptable to this algorithm with an appropriate regulation for each edge length (see [1]).

Table 3 displays the result of the application of this algorithm to seven sequences of proteins used in the previous section. The case that $k = 1$ corresponds to applying the A* algorithm and the score for this case is the optimal. The result shows the effectiveness of this algorithm. While both expanded vertices and visited vertices are reduced with a little larger k than unity, the score of the alignment decreases little. Particularly in the case $k = 1.001$, the optimal alignment is obtained in 40 seconds with only 965 vertices expanded. Although it is not certain that the obtained alignment is the optimal, enhanced A* efficiently finds the actual optimal alignment with the length of the path corresponding to the obtained alignment as an upper bound for the shortest path length.

5 Conclusion

In this paper, some algorithms based on the A* algorithm have been proposed for more than two dimensional multiple sequence alignment problem.

The first algorithm is the A* algorithm with the estimator utilizing the information on the result of all pairwise problems. The result of the experiment shows only few vertices are expanded in this algorithm while enormous amount of vertices conceptually exist in the graph. However, the number of visited vertices increases excessively as the dimension becomes higher. This phenomenon affects the efficiency of the algorithm since it increases the heap size and the necessary space. As for the heap size, such method as used in the bin sort is applicable to the search with the A* algorithm and can improve time complexity. This method will be implemented. On the other hand, the necessary space is essential to the A* algorithm and cannot be decreased in regular way.

The second algorithm overcomes this problem utilizing the upper bound of the shortest path length. This problem is due to the feature of the A* algorithm that it visits the all adjacent vertices to the vertex it expands. The second algorithm does not visit unnecessary vertices and keeps the necessary space sufficiently small. The same effect may be obtained with the information on all pairwise problems instead of an upper bound. This idea will be further shaped up in future.

The third algorithm is an approximate method based on the A algorithm, which provides the almost optimal alignment in extremely small time. This algorithm is useful not only as it is but also as the method to calculate an upper bound. An elegant measure for negative length edges and the criterion for the value of the parameter k remain open as future works.

Acknowledgment

This work was supported in part by the Grant-in-Aid on Priority Areas "Genome Informatics" of the Ministry of Education, Science and Culture of Japan.

References

[1] S. Araki, M. Goshima, S. Mori, H. Nakashima, S. Tomita, Y. Akiyama, and M. Kanehisa, "Application of Parallelized DP and A* Algorithm to Multiple Sequence Alignment," *Proc. Genome Informatics Workshop IV,* 1993, pp.94–102.

[2] P. E. Hart, N. J. Nillson, and B. Rafael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Trans. Sys. Sci. and Cyb. SSC-4,* 1968, pp.100–107.

[3] G. Shibayama and H. Imai, "Finding K-best Alignment of Multiple Sequences," *Proc. Genome Informatics Workshop IV,* 1993, pp.120–129.

[4] Y. Shirai and J. Tsuji, *Artificial Intelligence*, Iwanami Course: Information Science vol.22, Iwanami, Tokyo, 1982 (in Japanese).

[5] J. L. Spouge, "Speeding Up Dynamic Programming Algorithms for Finding Optimal Lattice Paths," *SIAM J. Appl. Math. 49,* 1989, pp.1552–1566.