VNU Journal of Science: Comp. Science & Com. Eng., Vol. 35, No. 1 (2019) 46-55

VNU Journal of Science:
Computer Science and Communication Engineering

Journal homepage: http://www.jcsce.vnu.edu.vn/index.php/jcsce

Original Article

Adaptive Large Neighborhood Search Enhances Global Protein-Protein Network Alignment

Vu Thi Ngoc Anh[1, 2], Nguyen Trong Dong[2],
Nguyen Vu Hoang Vuong[2], Dang Thanh Hai[3, *], Do Duc Dong[3, *]

[1]*The Hanoi college of Industrial Economics,*
[2]*VNU University of Engineering and Technology, 144 Xuan Thuy, Cau Giay, Hanoi, Vietnam,*
[3]*Bingo Biomedical Informatics Laboratory (Bingo Lab), Faculty of Information Technology, VNU University of Engineering and Technology*

Received 05 March 2018
Revised 19 May 2019; Accepted 27 May 2019

Abstract: Aligning protein-protein interaction networks from different species is a useful mechanism for figuring out orthologous proteins, predicting/verifying protein unknown functions or constructing evolutionary relationships. The network alignment problem is proved to be NP-hard, requiring exponential-time algorithms, which is not feasible for the fast growth of biological data. In this paper, we present a novel global protein-protein interaction network alignment algorithm, which is enhanced with an extended large neighborhood search heuristics. Evaluated on benchmark datasets of yeast, fly, human and worm, the proposed algorithm outperforms state-of-the-art algorithms. Furthermore, the complexity of ours is polynomial, thus being scalable to large biological networks in practice.

Keywords: Heuristic, Protein-protein interaction networks, network alignment, neighborhood search.

1. Introduction

Advanced high-throughput biotechnologies have been revealing numerous interactions between proteins at large-scales, for various species. Analyzing those networks is, thus, becoming emerged, such as network topology analyses [1], network module detection [2], evolutionary network pattern discovery [3] and network alignment [4], etc.

From biological perspectives, a good alignment between protein-protein networks (PPI) in different species could provide a strong evidence for (i) predicting unknown functions of orthologous proteins in a less-well studied species, or (ii) verifying those with known functions [5], or (iii) detecting common orthologous pathways between species [6] or (iv) reconstructing the evolutionary dynamics of various species [4].

PPI network alignment methods fall into two categories: local alignment and global alignment. The former aims identifying sub-networks that are conserved across networks in terms of topology and/or sequence similarity

* Corresponding author.
E-mail address: {hai.dang, dongdoduc}@vnu.edu.vn
https://doi.org/10.25073/2588-1086/vnucsce.228

[7-11]. Sub-networks within a single PPI network are very often returned as parts of local alignment, giving rise to ambiguity, as a protein may be matched with many proteins from another target network [12]. The latter, on the other hand, aims to align the whole networks, providing unambiguous one-to-one mappings between proteins of different networks [4, 12, 13-16].

The major challenging of network alignment is computational complexity. It becomes even more apparent as PPI networks are becoming larger (Network may be of up to 10^4 or even 10^5 interactions). Nevertheless, existing approaches are optimized only for either the performance accuracy or the run-time, but not for both as expected, for networks of medium sizes. In this paper, we introduce a new global PPI network (GPN) algorithms that exploit the adaptive large neighborhood search. Thorough experimental results indicate that our proposed algorithm could attain better performance of high accuracy in polynomial run-time when compared to other state-of-the-art algorithms.

2. Problem statement

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be PPI networks where V_1, V_2 denotes the sets of nodes corresponding to the proteins. E_1, E_2 denotes the sets of edges corresponding to the interactions between proteins. An alignment network A_{12}= (V_{12}, E_{12}), in which each node in V_{12} can be presented as a pair $< u_i, v_j >$ where $u_i \in V_1$, $v_j \in V_2$. Every two nodes $< u_i, v_j >$ and $< u'_i, v'_j >$ in V_{12} are distinct in case of $u_i \neq u'_i$ and $v_j \neq v'_j$. The edge set of alignment network are the so-called conserved edge, that is, for edge between two nodes $< u_i, v_j >$ and $< u'_i, v'_j >$ if and only if $< u_i, u'_i > \in E_1$ and $< v_j, v'_j > \in E_2$.

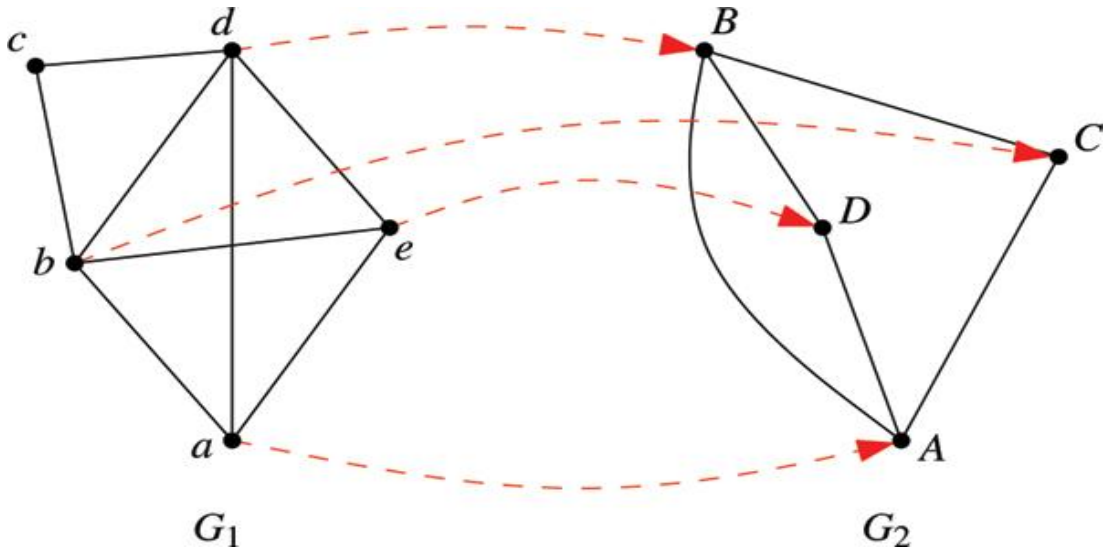

Figure 1. An example of an alignment of two networks [17].

Although an official definition of successful alignment network is not proposed, informally the common goal of recent approaches is to provide an alignment so that the edge set E_{12} is large and each pair of node mappings in the set V_{12} contains proteins with high sequence similarity [4, 18, 13, 14]. Formally, the definition of pairwise global PPI network alignment problem of A_{12} = (V_{12}, E_{12}) is to maximize the global network alignment score, defined as follows [12]:

$$GNAS(A_{12}) = \alpha \times |E_{12}| + (1 - \alpha) \times \sum_{\forall <u_i, v_j>} seq(u_i, v_j)$$

The constant $\alpha \in [0, 1]$ in this equation is a balancing parameter intended to vary the relative importance of the network-topological similarity (conserved edges) and the sequence similarities reflected in the second term of sum. Each $seq(u_i, v_j)$ can be an approximately defined sequence similarity score based on measures such as BLAST bit-scores or E-values.

3. Related state-of-the-art work

By far there have been various computational models proposed for global alignment of PPI networks (e.g. [4, 12, 13, 14, 15, 16], as alluded in the introduction section). Among them, to the best of our knowledge, Spinal and FastAN are recently state-of-the-art.

3.1. SPINAL

SPINAL, proposed by *Ahmet E. Aladağ* [12], is a polynomial runtime heuristic algorithm, consisting of two phases: Coarse-grained phase alignment phase and fine-grained alignment phase. The first phase constructs all pairwise initial similarity scores based on pairwise local neighborhood matching. Using the given similarity scores, the second phase builds one-to-one mapping bfy iteratively growing a local improvement subset. Both phases make use of the construction of neighborhood bipartite graphs and the contributors as a common primitive. SPINAL is tested on PPI networks of yeast, fly, human and worm, demonstrating that SPINAL yields better results than IsoRank of Singh et al. (2008) [13] in terms of common objectives and runtime.

3.2. FastAN

FastAN, proposed by *Dong* et al. (2016) [16], includes two phases, called Build and Rebuild. They both employ the same strategy similar to neighborhood search algorithms (see Section 4.1) that repeatedly destroy and repair the current found solution. The first phase is to build an initial global alignment solution by selecting iteratively an unaligned node from one network, which has the most connections to aligned nodes in the network, to pair with the best-matched node from the other network (See the Build phase, the first For loop, in Algorithm 1). The second phase follows the worst removal strategy to destroy the worst parts (99%) of the current solution based on their scores independently calculated. FastAN keeps 1% best pairs remained as a seeding set for reconstructing the solution. The reconstructing procedure is the same as the first phase. It reconstructs the destroyed solution by repeatedly adding best parts at the moment. FastAN accept every newly created solution from which it randomly choose one to follow. Using the same objective function and the dataset as SPINAL, FastAN yields much better result than SPINAL [12].

4. Materials

4.1. Neighborhood search

Given S the set of feasible solutions for globally aligning two networks and I being an instance (or input dataset) for the problem, we denote $S(I)$ when we need to emphasise the connection between the instance and solution set. Function $c: S \rightarrow \mathbb{R}$ maps from a solution to its cost. S is assumed to be finite, but is usually an extremely large set. We assume that the combinatorial optimization problem is a maximization problem, that is, we want to find a solution s^* such that $c(s^*) >= c(s) \; \forall s \in S$.

We define a neighborhood of a solution $s \in S$ as $N(s) \subseteq S$. That is, N is a function that maps a solution to a set of solutions. A solution s is considered as locally optimal or a local optimum with respect to a neighborhood N if $c(s) >= c(s') \; \forall s' \in N(s)$. With these definitions it is possible to define a neighborhood search algorithm. The algorithm takes an initial solution s as input. Then, it computes $s' = arg\, max_{s'' \in N(s)} \{c(s'')\}$, that

is, it searches the best solution s' in the neighborhood of s. If *c(s') > c(s)* is found, the algorithm performs an update $s = s'$. The neighborhood of the new solution s is continuously searched until it is converged in a region where local optimum s is reached. The local search algorithm stops when no improved solution is found (see Algorithm 1). This neighborhood search (NS), which always accepts a better solution to be expanded, is denoted a *steepest descent* (*Pisinger*) [19].

Algorithm 1. Neighborhood search in pseudo codes

```
INPUT: problem instance I
Create initial solution s_min ∈ S(I);
WHILE (stopping criteria not met) {
  s' = r(d(s));
    IF accept(s, s') {
      s = s';
        IF c(s') > c(s_min)
          s_min = s';
    }
}
return s_min
```

4.2. Large neighborhood search

Large neighborhood search (LNS) was originally introduced by *Shaw* [20]. It is a meta-heuristic that neighborhood is defined implicitly by a destroy-and-repair function. A destroy function destructs part of the current solution s while repair function rebuilds the destroyed solution. The destroy function should pre-define a parameter, which controls the degree of destruction. The neighborhood $N(s)$ of a solution s is calculated by applying the destroy-and-repair function.

4.3. Adaptive Large Neighborhood search

Adaptive Large Neighborhood Search (ALNS) is an extension of Large Neighborhood Search and was proposed by *Ropke and Prisinger* [19]. Naturally, different instances of an optimization problem are handled by different destroy and repair functions with varying level of success. It may difficult to decide which heuristics are used to yield the best result in each instance. Therefore, ALNS enables user to select as many heuristics as he wants. The algorithm firstly assigns for each heuristic a weight which reflects the probability of success. The idea, that passing success is also a future success, is applied. During the runtime, these weights are adjusted periodically every P_u iterations. The selection of heuristics based on its weights. Let $D = \{d_i \,|i = 1..k\}$ and $R = \{r_i \,|i = 1..l\}$ are sets of destroy heuristics and repair heuristics. The weights of heuristics are $w(r_i)$ and $w(d_i)$. $w(r_i)$ and $w(d_i)$ are initially set as 1, so the probability of selection of heuristics are:

$$p(r_i) = \frac{w(r_i)}{\sum_{j=1}^{l} w(r_j)} \text{ and } p(d_i) = \frac{w(d_i)}{\sum_{j=1}^{k} w(d_j)}$$

Apart from the choice of the destroy-and-repair heuristics and weight adjustment every update period, the basic structure of ALNS is similar LNS (see Algorithm 2).

Algorithm 2: Adaptive Large Neighborhood Search algorithm

```
INPUT: problem instance I
Create initial solution s_min ∈ S(I);
WHILE (stopping criteria not met) {
  FOR i = 1 TO p_u DO {
    select r ∈ R, d ∈ D according to probability;
    s' = r(d(s));
    IF accept(s, s') {
      s = s';
      IF c(s') > c(s_min)
        s_min = s';
    }
    update weight w, and probability p;
}return s_min
```

5. Proposed model

We note that FastAN still has some limitations, including: (i) randomly choosing a

newly constructed solution to follow may yield the unexpected results, gearing to the local optimum by chance. (ii) The fixed degree of destruction at 99% may reduce the flexibility of neighborhood searching process. Setting this degree too large can be used to diverse the search space, however, would cause the best results hardly to be reached. Newly constructed solutions are not real neighbors of the current solution, thus being totally irrelevant solutions). (iii) The heuristic worst part removal of the current solution may get FastAN stuck in a local optimum because of the absence of diversity. Moreover, using only one heuristic does not guarantee the best result found for different instances of problem. (iv) The basic greedy heuristic in ALNS is employed to repair destroyed solutions. Although it always guarantees better solutions to be yielded, but it is not the optimal way to construct the best solution. There is another better heuristic called n-regret could be employed. (v) Using only one destroy heuristic and one repair (construction) heuristic does not provide the weight adjustment. Two heuristics are always chosen with 100% of probability.

To this end, in this paper, we aim at eliminating those limitations by proposing a novel global protein-protein network alignment model that is mainly based on FastAN. Unlike FastAN, which employs a neighborhood search algorithm, the proposed model improves FastAN by adopting a rigorous adaptive large neighborhood search (ALNS) strategy for the second phase (namely Rebuild) of FastAN. The Build phase is similar to that of FastAN (See Alogrithm 3).

Alogrithm 3: Pseudo code for our proposed PPI alignment algorithm

INPUT: $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$,
Similarity Score Seq[i][j], balance factor α
OUTPUT: An alignment A_{12}
//Build Phase, similar to that of FastAN [21]
$V_{12} = < i, j >$ //with seq[i][j] is maximum
FOR $k = 2$ ***TO*** $|V_1|$ ***DO*** {
  $i = find\_next\_node(G_1)$;
  $j = find\_best\_match(i, G_1, G_2)$;
  $V_{12} = V_{12} \cap < i, j >$;
}
//Rebuild phase
FOR $iter = 1$ ***TO*** n_iter ***DO*** {
  $d = get\_d(d_{min}, d_{max})$;
  $destroy\_heuristic = select\_destroy\_heuristic()$;
  $repair\_heuristic = select\_repair\_heuristic()$;
  $new\_sol = destroy(destroy\_heuristic, V_{12}, d)$;
  $new\_sol = repair(repair\_heuristic, new\_sol)$;
  //reward for successful heuristics
  IF $(G\_BEST < score(new\_sol))$ {
    $G\_BEST = score(new\_sol)$;
    $reward(destroy\_heuristic, repair\_heuristic, \delta_1)$;
  }
  IF $(score(V_{12}) < score(new\_sol))$
    $reward(destroy\_heuristic, repair\_heuristic, \delta_2)$;
  IF $(accept(V_{12}, new\_sol))$ {
    $V_{12} = new\_sol$;
    $reward(destroy\_heuristic, repair\_heuristic, \delta_3)$;
  }
  IF $(iter \,\%\, update\_period == 0)$
    weight_$adjustment()$;
}
return V_{12};

The proposed algorithm uses a simple Threshold Acceptance (TA) heuristic for adaptive large neighborhood search. TA accepts any solutions of which its difference from the best so far (G-BEST) is not greater than T, a manually given parameter in range [0, positive inf) (see Procedure 1).

Procedure 1. Accept function used for adaptive large neighborhood search

Boolean accept_function (sol, new_sol) {
  IF $(cost_{sol} - cost_{new\_sol} \leq T)$
    return $True$;
  return $False$;
}

Note that the threshold T is set as a constant rather than increasing or decreasing due to the

success of heuristic. The algorithm is supposed to search around the G_BEST solution at a constant radius. Decreasing the radius may limit the search space due to the fact that there are still many other heuristics, which have a chance to find better results.

The degree of destruction used in our ALNS of the proposed algorithm has the opposite meaning: in particular, d is the size of seeding set, not the destruction degree (see the second For loop in Algorithm 3). d is randomly selected from the range $[d_{min}, d_{max}]$, two given parameters of the algorithm. The suggested range is from 0.01 to 0.1; meaning that the algorithm should destroy 90% to 99% the solution.

There are two destroy heuristics for ALNS in our proposed algorithm, namely Random Removal and Worst Removal. The former destroys the current solution at some randomly chosen part of the solution while the latter at the worst part. It is argued that Worst Removal is better than Random removal in term of yielding better local result, but lack of randomization. The combination of Random Walk and Worst Removal is suggested to deal with this problem. It raises a concern that Random Removal may not yield the best result; however, it does not happen due to the observation that the probability of choice Random Walk always decreases after a few iterations. As a result, this heuristic is not often selected and does not touch the solution quality rebuild process. Nevertheless, Random Walk contributes to diverse search space, which solves the drawback of Worst Removal.

Regarding the repair heuristic in ALNS of the proposed algorithm, we proposed two heuristics, i.e. Basic Greedy and n-regret. Basic Greedy heuristic is same as that in FastAN. The difference is the n-regret heuristic (see Procedure 2), in which we selected the top 3 best candidates from V_1 that have the most connections to the seeding set. Of course, these candidates have had to not appear in the seeding set yet. The next steps is that we loop every candidate from V_2 calculate the best and second-best score of each pairs. Candidate from V_2 should not appear in seeding set also. The candidate, from V_1 that has biggest gap from its best and second best, is selected. The corresponding candidate V_2 is also selected.

Procedure 2: n_regret heuristic in pseudo codes

```
Solution n_regret(seeding_set) {
  WHILE seeding_set is not full {
    top_3 = {};

  FOR every u in V1 but not in seeding_set {

  IF (connections_to_seeding_set(u, seeding_set
        update top_3;
    }
    diff_1 = diff_2 = diff_3 = 0;

  FOR every v in V2 but not in seeding_set {
      Calculate best_u1, best_u2, best_u3;
      Calculate second_best_u1, second_best_u2,
                          second_best_u3;
      diff_1 = |best_u1 – second_best_u1|;
      diff_2 = |best_u2 – second_best_u3|;
      diff_3 = |best_u3 – second_best_u3|;
    }

  select candidate which has biggest diff denot
          as (candV1, candV2);

  add (candV1, candV2) pair to seeding_set;
  }
  return seeding_set;
}
```

It can be seen that, 1_regret is Basic Greedy which always select the candidate from V_1 which has the most connections and the best score from the candidate from V_2. An obvious problem of Basic Greedy is that it often postpones the placement of *difficult choice* to the last iterations where we do not have much freedom of action. The regret heuristic tries to circumvent the problem by incorporating a kind of look-ahead information when selecting the request to insert. The Regret heuristic had been used by *Potvin and Rousseau* [21] for the VRPTW and in the context of the generalized assignment problem *Trick* [22].

Let Δf_u^q be the change in the objective value incurred by adding pair u, v, which v is the q^{th} candidate from V_2 corresponding to u, to the seeding-set. For example Δf_u^2 denote the change when adding pair u, and its second-best v. Each selection, the regret heuristic chooses to insert u according to:

$$u = \arg max_{u\ in\ V_1} \left(\sum_{h=2}^{n} \Delta f_u^1 - \Delta f_u^h \right)$$

The candidate u is selected with a maximum the cost of v. It means that we maximize the difference of cost of selecting candidate u in its best way and its second best way. Ties can be broken by randomly choosing among them. The proposed algorithm repeats until *seeding_set* is full. Clearly, higher n, longer the run time, so that the regret heuristic is used in the new algorithm is 2-regret heuristic. Also, the set V_1 and V_2 are up to $1e4$, so that we can not consider all candidate from V_1, that explains why top 3 candidate u from V_1 are chosen to applying regret strategy.

The proposed algorithm uses the weight adjustment strategy for ALNS, which is as the same as that in [22]. As we mentioned above, the weight of Random Walk are always much lower than that of Worst Removal, and quickly decreases to 0. All weights are set at 1 initially. Interestingly, the weights of *n_regret* always outperform those of Basic Greedy, so that the properties of *n_regret* are strongly convinced. The Worst Removal heuristic, however, is not too low at all. It means that Worst Removal is still a good heuristic in network alignment problem.

6. Experimental results

6.1. Implementation and datasets

Our proposed algorithm is implemented in C++11; source code is freely available at https://github.com/meodorewan/thesis. We do experiments on benchmark data sets from four species: Saccharomyces cerevisiae, Drosophila melanogaster, Caenorhabditis elegans and Homo sapiens. All datasets are used in all state-of-the-art models, i.e. IsoRank, SPINAL, FastAN, etc. The PPI network sizes are as follows: 5499 proteins and 31 261 interactions in the S. cerevisiae network, (7518, 25 635) in D. melanogaster, (2805, 4495) in C. elegans and (9633, 34327) in H. sapiens (Table 1).

Table 1. Number of proteins and interactions between them in experimental datasets

Dataset	Number of Proteins	Number of Interactions
Saccharomyces cerevisiae	5499	31261
Drosophila melanogaster	7518	25635
Caenorhabditis elegans	2805	4495
Homo sapiens	9633	34327

6.2. Experimental results in comparison with FastAN

We first examine the efficiency of each improvement in the proposed algorithm including strategy of choosing a degree of destruction, different destroy and repair functions. The objective function is described in section 1.2. Results for each improvement are compared with those of FastAN.

6.3. Improvement with randomization of destruction degree

Here is the first improvement, we keep all settings as same as the original FastAN algorithm except for only the strategy of choosing d. FastAN is using destroy heuristic Worst Removal, and repair heuristic is Basic Greedy. It fixed d = 99%, while we randomize parameter d in range $[d_{min}, d_{max}]$.

Table 2. Experimental results of FastAN + d.

Dataset	α = 0.3		α = 0.5		α = 0.7	
	FastAN	FastAN + d	FastAN	FastAN + d	FastAN	FastAN + d
ce-dm	*778.46*	***823.19***	*1290.11*	***1363.42***	*1801.24*	***1915.25***
ce-hs	*863.46*	***878.79***	*1429.89*	***1445.54***	*1994.87*	***2035.78***
ce-sc	*834.79*	***867.58***	*1389.21*	***1434.13***	*1936.83*	***2016.16***
dm-hs	*2260.31*	***2318.82***	*3755.36*	***3857.11***	*5242.32*	***5402.33***
dm-sc	*1977.82*	***2020.35***	*3290.03*	***3361.21***	*4603.41*	***4688.87***

hs-sc	*2268.21*	***2342.29***	*3772.96*	***3911.03***	*5279.88*	***5444.05***

Through the experimental results shown in Table 2, we can conclude that the strategy of choosing destruction degree is advantaged. The results are much better than that of original FastAN with fixed d at 99%. The reason is that fixed parameter d may limit the search space and be difficult to find a new local optimum. By randomizing d in range $[d_{min}, d_{max}]$, we can diverse the neighborhoods and be able to find better optimum.

6.4. Improvement with destroy heuristic Random Removal

Setting of this improvement is that we use one destroy heuristic (i.e. Random Removal) instead of the Worst Removal in FastAN. Other settings are kept, including destruction degree at 99% for the repair heuristic (Basic Greedy). Experiment shown in Table 3 demonstrates that destroy heuristic Random Removal is disoriented searching strategy, it can be useful when local minimum reached, but disadvantaged during searching process. This explains why we should set the weight of this heuristic much lower than other oriented searching strategies.

Table 3. Experimental results of FastAN + random removal.

Dataset	$\alpha = 0.3$		$\alpha = 0.5$		$\alpha = 0.7$	
	FastAN	FastAN + RR	FastAN	FastAN + RR	FastAN	FastAN + RR
ce-dm	***778.46***	*733.57*	***1290.11***	*1211.63*	***1801.24***	*1680.53*
ce-hs	***863.46***	*816.59*	***1429.89***	*1351.99*	***1994.87***	*1889.16*
ce-sc	***834.79***	*790.07*	***1389.21***	*1307.96*	***1936.83***	*1831.65*
dm-hs	***2260.31***	*2109.93*	***3755.36***	*3498.53*	***5242.32***	*4886.54*
dm-sc	***1977.82***	*1837.01*	***3290.03***	*3056.96*	***4603.41***	*4272.97*
hs-sc	***2268.21***	*2092.27*	***3772.96***	*3476.05*	***5279.88***	*4890.21*

6.5. Improvement with repair heuristic 2-regret

Setting of this improvement is about repair heuristic. We examine the efficiency of the 2-regret heuristic comparing to Basic Greedy one. All other settings are kept originally. The result shows that the *2-regret* heuristic outperformed most of the tests except ce-hs one (Table 4). It can be concluded that the heuristic *2-regret* is better than Greedy heuristic in most of the cases.

Table 4. Experimental results of FastAN + *2-regret* repair heuristic.

Dataset	$\alpha = 0.3$		$\alpha = 0.5$		$\alpha = 0.7$	
	FastAN	FastAN + regret-2	FastAN	FastAN + regret-2	FastAN	FastAN + regret-2
Ce-dm	*778.46*	***815.99***	*1290.11*	***1352.25***	*1801.24*	***1881.70***
ce-hs	***863.46***	*860.24*	***1429.89***	*1413.04*	***1994.87***	*1965.16*
ce-sc	*834.79*	***864.33***	*1389.21*	***1429.55***	*1936.83*	***2007.28***
dm-hs	*226031*	***2281.21***	*3755.36*	***3788.08***	*5242.32*	***5290.47***
dm-sc	*1977.82*	***1983.21***	*3290.03*	***3297.65***	*4603.41*	***4603.61***
hs-sc	*2268.21*	***2274.16***	*3772.96*	***3784.53***	*5279.88*	***5283.64***

6.6. Improvement with the adaptive framework

In this version, we applied the adaptive strategy without modification of destruction degree. In other words, this version is similar to the new algorithm except for fixed destruction degree at 99%. This version is to compare the efficiency of an adaptive framework with original FastAN algorithm. The experiment results reveal that adaptive framework works better in three smaller tests, but not effective in three large ones (Table 5). It can be explained that local optimum is not reached, we should increase the number of iterations to get better results than those of FastAN.

Table 5: Experimental results of FastAN + adaptive framework.

Dataset	$\alpha = 0.3$		$\alpha = 0.5$		$\alpha = 0.7$	
	FastAN	FastAN + adaptive	FastAN	FastAN + adaptive	FastAN	FastAN + adaptive
ce-dm	*778.46*	***783.815***	*1290.11*	***1310.45***	*1801.24*	***1812.91***
ce-hs	*863.46*	***875.09***	*1429.89*	***1453.00***	*1994.87*	***2018.28***
ce-sc	*834.79*	***841.13***	*1389.21*	***1408.47***	*1936.83*	***1950.30***
dm-hs	*2260.31*	*2208.78*	*3755.36*	*3646.98*	*5242.32*	*5099.03*
dm-sc	*1977.82*	*1920.44*	*3290.03*	*3195.56*	*4603.41*	*4467.44*
hs-sc	*2268.21*	*2231.89*	*3772.96*	*3691.48*	*5279.88*	*5177.50*

Table 6. Parameters settings of the proposed algorithm

Parameter	Describe	Setting
d_{min}	The lower bound of degree of destruction	0.01
d_{max}	The upper bound of degee of destruction	0.1
N_RUN	The number of iteration	100
PERIOD	The update period for weight adjustment	5
ρ	The degenerative factor	0.1
δ_1	Reward for solution which has best cost so far	0.8
δ_2	Reward for solution which has better cost	0.3
δ_3	Reward for solution which is accepted	0
N_TEST	Number of execution to test the stability of algorithm	10
T	Threshold	5

6.7. Results in terms of alignment objectives

We measure the accuracy of the proposed algorithms in terms of the maximization objective formulated in section 1.2. The number of conserved interactions, that is, the edge set size of the alignment network, denoted with E_{12} in the equation is a common performance indicator used in almost all the global network alignment studies [4, 18, 13, 14]. Because the optimization goal is also commonly defined as in section 1.2, we include the score obtained from $GNAS(A_{12})$ as well as $|E_{12}|$ in our evaluations of an alignment A_{12}. The studied algorithms are examined under a specific setting of input parameters. Parameter setting for the proposed algorithm consists of varying the constant α from 0.3 to 0.7 in the increments of 0.2 (see Table 6 for other settings). Table 7 summarizes the performance in terms of such two objectives of the proposed algorithms in comparison with SPINAL and FastAN. Obviously, the new algorithm yields the highest scores for all datasets examined.

6.8. Complexity and runtime

The complexity of the proposed algorithm is same as FastAN $O(|V_1| * |E_1| + |V_1| * |E_2|)$ for each iteration. The number of iteration is constant. All additional heuristics used have the

Table 7. Performance in terms of two objectives (i.e. the size of conserved interactions set E_{12} and the bottom indicates the score obtained from $GNAS(A_{12})$) of the proposed algorithms (indicated by "Ours") in comparison with SPINAL and FastAN.

Dataset	$\alpha = 0.3$			$\alpha = 0.5$			$\alpha = 0.7$		
	SPINAL	FastAN	Ours	SPINAL	FastAN	Ours	SPINAL	FastAN	Ours
ce-dm	*717.99*	*778.46*	***821.98***	*1159.93*	*1290.11*	***1348.1***	*1586.87*	*1801.24*	***1885.1***
	2343	*2560.7*	***2710.8***	*2300.0*	*2567.2*	***2684.9***	*2258.0*	*2567.6*	***2688.4***
ce-hs	*728.26*	*863.46*	***913.59***	*1229.95*	*1429.89*	***1482.3***	*1764.93*	*1994.87*	***2061.8***
	2370	*2842.8*	***3016.1***	*2437.0*	*2844.9*	***2952.8***	*2512.0*	*2843.4*	***2940.3***
ce-sc	*709.12*	*834.79*	***884.48***	*1168.95*	*1389.21*	***1454.9***	*1683.13*	*1936.83*	***2023.4***
	2326	*2761.1*	***2930.9***	*2323.0*	*2769.7*	***2902.6***	*2398.0*	*2763.1*	***2887.6***
dm-hs	*1883.22*	*2260.31*	***2305.2***	*3160.48*	*3755.36*	***3785.5***	*4451.6*	*5242.32*	***5285.9***
	6189	*6569.7*	***7633.7***	*6282.0*	*7429.0*	***7549.6***	*6344.0*	*7478.8*	***7542.2***
dm-sc	*1579.06*	*1977.82*	***2017.5***	*2668.65*	*3290.03*	***3346.0***	*3759.07*	*4603.41*	***4657.6***
	5203	*6569.7*	***6702.6***	*5311.0*	*6570.7*	***6682.7***	*5360.0*	*6572.3*	***6649.7***
hs-sc	*1731.81*	*2268.21*	***2302.4***	*2839.00*	*3772.96*	***3869.0***	*4066.22*	*5279.88*	***5383.5***
	5703	*7531.8*	***7648.7***	*5651.0*	*7535.2*	***7728.4***	*5798.0*	*7538.1*	***7686.6***

same complexity as it is in Rebuild phase. The proposed algorithm's runtime is also same as FastAN's runtime.

The hardware used to run the experiment is an Intel(R) Xeon(R) CPU E5-2697 v4 @ 2.30GHz 16GB of RAM. Comparison runtime is shown below. The runtime of the new algorithms is likely to be as three times as that of FastAN and approximately equal to SPINAL's runtime with all size of datasets (see Table 8). This can be explained that the complexity of constant multiply depends on which heuristic is selected. For example, the complexity constant multiply for 2-regret repair heuristic is 3. However, it has no meaning for complexity analysis.

Table 8. Runtime of the proposed algorithm in comparison with SPINAL and FastAN.

Dataset	SPINAL	FastAN	New algorithm
ce-dm	*540.2*	*221.5*	*697.9*
ce-hs	*664.3*	*327.9*	*846.6*
ce-sc	*638.2*	*142.2*	*588.4*
dm-hs	*1736.8*	*1395.9*	*3924.4*
dm-sc	*1912.1*	*1064.5*	*2238.8*
hs-sc	*2630.6*	*1507.8*	*2497.6*

7. Discussion and future work

In this paper we proposed a novel global protein-protein network alignment algorithm, which is mainly based on FastAN algorithm [16]. Ours improves FastAN by applying the Adaptive Large Neighborhood Search. We have solved several limitations of FastAN by proposing two destroy/repair heuristics, and a new accept a function as well. Thorough experiments demonstrate out-performance of the proposed algorithm when compared to FastAN. We note that the parameters used in the proposed algorithm have not been tuned yet. Tuning them can be a potential for further perspective work.

Acknowledgments

This work has been supported by VNU University of Engineering and Technology under project number **CN18.19**.

References

[1] J.D. Han et al, Evidence for dynamically organized modularity in the yeast proteinprotein interaction network, Nature. 430 (2004) 88-93.

[2] G.D. Bader, C.W. Hogue, Analyzing yeast protein-protein interaction data obtained from different sources, Nat. Biotechnol. 20 (2002) 991-997.

[3] H.B. Hunter et al, Evolutionary rate in the protein interaction network, Science. 296 (2002) 750-752.

[4] O. Kuchaiev, N. Prz˘ ulj, Integrative network alignment reveals large regions of global network similarity in yeast and human, Bioinformatics. 27 (2011) 1390-1396.

[5] J. Dutkowski, J. Tiuryn, Identification of functional modules from conserved ancestral protein-protein interactions, Bioinformatics. 23 (2007) i149-i158.

[6] B.P. Kelley et al, Conserved pathways within bacteria and yeast as revealed by global protein network alignment, Proc. Natl Acad. Sci. USA. 100 (2003) 11394-11399.

[7] B.P. Kelley et al, Pathblast: a tool for alignment of protein interaction networks, Nucleic Acids Res. 32 (2004) 83-88.

[8] R. Sharan et al, Conserved patterns of protein interaction in multiple species, Proc. Natl Acad. Sci. USA. 102 (2005) 1974-1979.

[9] M. Koyuturk et al, Pairwise alignment of protein interaction networks, J. Comput. Biol. 13 (2006) 182-199.

[10] M. Narayanan, R.M. Karp, Comparing protein interaction networks via a graph match-and-split algorithm, J. Comput. Biol. 14 (2007) 892-907.

[11] J. Flannick et al, Graemlin: general and robust alignment of multiple large interaction networks, Genome Res. 16 (2006) 1169-1181.

[12] E. hmet, Aladağ, Cesim Erten, SPINAL: scalable protein interaction network alignment, Bioinformatics. Volume 29(7) (2013) 917-924. https://doi.org/10.1093/bioinformatics/btt071.

[13] R. Singh et al, Global alignment of multiple protein interaction networks. In: Pacific Symposium on Biocomputing, 2008, pp. 303-314.

[14] M. Zaslavskiy et al, Global alignment of protein-protein interaction networks by graph matching methods, Bioinformatics. 25 (2009) 259-267.

[15] L. Chindelevitch, Extracting information from biological networks. PhD Thesis, Department of Mathematics, Massachusetts Institute of Technology, Cambridge, 2010.

[16] Do Duc Dong et al, An efficient algorithm for global alignment of protein-protein interaction networks, Proceeding of ATC15, 2015, pp. 332-336.

[17] G.W. Klau et al, A new graph-based method for pair wise global network alignment, BMC Bioinformatics, (APBC 2009), 10(1), S59.

[18] L. Chindelevitch et al, Local optimization for global alignment of protein interaction networks, In: Pacific Symposium on Biocomputing, Hawaii, USA, 2010, pp. 123-132.

[19] S. Ropke, D. Pisinger, An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. Transportation Science. 40 (2006) 455-472. https:// doi.org/10.1287/trsc.1050.0135.

[20] P. Shaw, A new local search algorithm providing high quality solutions to vehicle routing problems, Technical report, Department of Computer Science, University of Strathclyde, Scotland, 1997.

[21] J.Y. Potvin, M. Rousseau, Parallel Route Building Algorithm for the Vehicle Routing and Scheduling Problem with Time Windows, European Journal of Operational Research. 66(3) (1993) pp. 331-340.

[22] M.A. Trick, A linear relaxation heuristic for the generalized assignment problem, Naval Research Logistics. 39 (1992) 137-151.