BIOINFORMATICS ORIGINAL PAPER

Vol. 30 no. 14 2014, pages 2043–2050
doi:10.1093/bioinformatics/btu150

Systems biology

Advance Access publication March 17, 2014

Optimal metabolic route search based on atom mappings

Mario Latendresse*, Markus Krummenacker and Peter D. Karp

Bioinformatics Research Group/Artificial Intelligence Center, SRI International, Menlo Park, CA 94025, USA

Associate Editor: Alfonso Valencia

ABSTRACT

Motivation: A key computational problem in metabolic engineering is finding efficient metabolic routes from a source to a target compound in genome-scale reaction networks, potentially considering the addition of new reactions. Efficiency can be based on many factors, such as route lengths, atoms conserved and the number of new reactions, and the new enzymes to catalyze them, added to the route. Fast algorithms are needed to systematically search these large genome-scale reaction networks.

Results: We present the algorithm used in the new RouteSearch tool within the Pathway Tools software. This algorithm is based on a general Branch-and-Bound search and involves constructing a network of atom mappings to facilitate efficient searching. As far as we know, it is the first published algorithm that finds guaranteed optimal routes where atom conservation is part of the optimality criteria. RouteSearch includes a graphical user interface that speeds user understanding of its search results. We evaluated the algorithm on five example metabolic-engineering problems from the literature; for one problem the published solution was equivalent to the optimal route found by RouteSearch; for the remaining four problems, RouteSearch found the published solution as one of its best-scored solutions. These problems were each solved in less than 5 s of computational time.

Availability and implementation: RouteSearch is accessible at BioCyc.org by using the menu command Metabolism → Metabolic RouteSearch and by downloading Pathway Tools. Pathway Tools software is freely available to academic users, and for a fee to commercial users. Download from: http://biocyc.org/download.shtml.

Contact: mario.latendresse@sri.com

Supplementary information: Supplementary data are available at *Bioinformatics* online.

Received on December 16, 2013; revised on February 14, 2014; accepted on March 11, 2014

1 INTRODUCTION

Metabolic engineers seek to modify the metabolic network of an organism to efficiently produce novel chemical products, with applications including the production of biofuels and specialized chemicals. The space of potential modifications is a large one given that metabolic databases such as MetaCyc (Caspi *et al.*, 2012) and KEGG (Kanehisa *et al.*, 2012) contain thousands of enzyme-catalyzed reactions that could be engineered into an organism. How do we balance the cost of engineering the organism against the efficiency of the resulting pathways?

The methods we present assume that the users begin with a source (feedstock) compound and a target (product) compound and that they aim to find the optimal metabolic routes that connect those compounds within a genome-scale metabolic reaction network. The methods further assume that the reaction network is annotated with atom mappings, which describe the path of each atom in a metabolic reaction from a reactant compound to a product compound. The methods allow the starting reaction network of the organism to be supplemented with additional reactions from an external reaction library, which for our RouteSearch program is the MetaCyc database.

RouteSearch searches for multiple optimal routes where optimality is essentially based on minimizing the number of native reactions of the route, the number of non-native reactions introduced in the route from the external library and the number of atoms lost from the start compound to the target compound according to the atom mappings of the reactions. We believe that these optimality criteria, in particular reducing the loss of atoms from the start compound to the target compound, are likely to provide good route candidates for metabolic engineering.

RouteSearch can also aid scientists in solving a simpler problem than the engineering of new metabolic routes, namely, the understanding of metabolic routes in existing metabolic networks. Scientists want to explore alternative routes within an organism's natural metabolic network for transforming a given starting compound into a product compound.

We present the algorithm used by the web-based interactive tool RouteSearch to find optimal metabolic routes. Optimality is based on the number of reactions used, the provenance of the reactions (natural or engineered) and the atoms conserved by the route from the source compound to the target compound. As far as we know, it is the first published exact algorithm that guarantees optimality of the routes found based, in part, on the number of atoms conserved from the source to the target compound.

RouteSearch is implemented in Common Lisp and is available at the BioCyc.org Web site for any of the 3000 organism-specific metabolic networks within BioCyc by using the menu bar command Metabolism → Metabolic RouteSearch, but without use of an external reaction library. MetaCyc can be used as the reaction library by downloading Pathway Tools (Karp *et al.*, 2010) and running it as a local web server.

Sections 2.1 and 2.2 present the RouteSearch search algorithm, which is based on the Branch-and-Bound algorithm applied to a precompiled metabolic network graph that we call a mapped network. The efficiency of the algorithm is discussed, as is its method of using atom mappings to trace atoms conserved in metabolic routes. The atom-mapping data used were

*To whom correspondence should be addressed.

computed by a technique presented in Latendresse *et al.* (2012) and applied to the MetaCyc (Caspi *et al.*, 2012) multi-organism database (version 17.5) to produce atom mappings for >10 300 biochemical reactions. These atom mappings are used for all other databases available at the BioCyc.org Web site.

Section 3 describes the RouteSearch graphical interface, which allows users to specify and control RouteSearch searches and speeds user understanding of the paths found by RouteSearch, including the atoms that are conserved from source to target compound.

Section 4 presents an empirical validation of RouteSearch on five example metabolic-engineering problems from the literature. For three problems the published solution was equivalent to the optimal route found by RouteSearch; for the remaining two problems the published solution was found by RouteSearch, but was not the optimal solution.

Section 4.2 compares RouteSearch with previous work on metabolic route-searching algorithms.

For efficiency, the metabolic route searches are done on mapped networks that are constructed using compounds, reactions and atom mappings. Their construction is explained in the following section.

2 ALGORITHM

2.1 Mapped networks

The searching algorithm presented in Section 2.2 uses a data structure called a *mapped network* and is constructed in a such a way to efficiently apply a Branch-and-Bound algorithm. Mapped networks can be reused across multiple searches. A mapped network is a network where vertices are compounds and arcs are reactions with their atom mappings. No compound is repeated in the mapped network, that is, a one-to-one correspondence exists between the vertices of the network and the compounds. On the other hand, a reaction might occur in several arcs, as it may modify more than one compound. Multiple arcs may exist between two compounds, because more than one reaction may modify one compound into another.

When RouteSearch is applied to an organism-specific metabolic network (e.g. EcoCyc), plus an external reaction library (e.g. MetaCyc), all the compounds and reactions involved in these two networks are combined in the mapped network, without duplication. In such a case, a *native* database (e.g. EcoCyc) and a database identified as a *library* of compounds and reactions (e.g. MetaCyc) exist. If a compound or reaction exists in the native and library databases, then the compound or reaction of the library is not included in the mapped network because the native compounds and reactions are assumed to take precedence. In summary, the library of compounds and reactions, in our case MetaCyc, is a resource to expand the native database.

As previously mentioned, the atom mappings of reactions were computed for the MetaCyc database as described in Latendresse *et al.* (2012). An atom mapping of a reaction is a bijection, that is, a function which is injective and surjective, of the reactant atoms to the product atoms. These atom mappings track all atom species except hydrogen. For any other Pathway/Genome Databases (PGDBs; e.g. EcoCyc), the atom mappings of its reactions are taken from MetaCyc. That is, we always rely on the atom mappings stored in MetaCyc to get the atom mappings for reactions coming from other PGDBs because MetaCyc contains a superset of the reactions in individual PGDBs.

More precisely, a mapped network is a directed graph $N(V, A)$ where V is a set of vertices and A is a set of arcs. Each vertex corresponds to one compound. An arc from compound v to compound u is a triple (v, u, R) where R is a set of pairs (r, M) where r is a unidirectional reaction and M is the atom mapping of r *restricted* from compound v to compound u. It is restricted in the sense that the atom mapping M applies only to the atoms of v that are mapped to u. The mapping M, which is a function, is implemented as a vector of integers M where $M[i] = j$ means that atom i of u maps to atom j of v. The reaction r is represented by the reactants and products involved in the reaction, that is, two sets of vertices. A reaction that is reversible is represented by two unidirectional reactions in the network, one for each direction.

Concretely, the set of all arcs between two compounds is partitioned into sets based on the atom mapping vectors M: reactions with the same restricted atom mapping M are in the same set. This representation increases the speed of searching for the optimal routes because all reactions in the same set only need one evaluation for the cost of atoms lost and conserved.

During the computation of the atom mappings, the following two cases occurred sometimes:

(1) For some reactions, several atom mappings were computed for each one and it was not possible to decide whether any of these multiple atom mappings were incorrect and they were not found to be equivalent. For such cases, all of the mappings were kept in MetaCyc. Less than 5% of reactions in MetaCyc have more than one atom mapping.

(2) For some reactions, several atom mappings were computed, but it was found that they were equivalent because of the symmetry of compounds, indistinguishable atoms or the duplication of a compound as reactants or products. Only one atom mapping was kept in MetaCyc as a representative of these equivalent atom mappings.

Note that case 1 is because of a lack of precision in the computation of the atom mappings, as we expect only one atom mapping to exist for one reaction, but the atom mapping algorithm cannot determine which is the correct mapping. For case 1, two subcases need to be considered for a specific pair of compounds c_1 and c_2 involved in the atom mappings of one reaction:

1a All atom mappings map either the same set of atoms from c_1 to c_2, or at least one atom mapping maps a superset of atoms from c_1 to c_2.

1b At least two atom mappings map different sets of atoms from c_1 to c_2 and neither being a superset of atoms over c_2.

In subcase 1a, if all atom mappings map the same set of atoms to c_2, then they are equivalent. If one atom mapping is a superset, then selecting that atom mapping is the correct choice because the optimality criteria defined in Section 2.2 maximizes the number of atoms conserved. For example, the Supplementary Figure S1 shows a reaction with two atom mappings, one mapping to a superset of the other, depending on the compounds considered.

Subcase 1b is more complex because, among other things, we would need to take into account the symmetry and stoichiometry of compounds. For example, Supplementary Figure S2 shows a reaction with two atom mappings, but the difference in atoms mapped is because of the symmetry of compound dihydroxyfumarate.

In case 2, only one atom mapping was kept that corresponds to several equivalent atom mappings. For example, Supplementary Figure S3 shows an atom mapping where two 3-phospho-D-glycerate compounds are involved: one transferring no atom to carbon dioxide, and the other one creating carbon dioxide; and where D-ribulose-1,5-bisphosphate is entirely formed by 3-phospho-D-glycerate. To fully account for these symmetries, it would be necessary to detect them for any compound and to assign probabilities of transfers of atoms depending on the type of symmetry involved. Moreover, stoichiometry would need to be taken into account. RouteSearch does not currently handle these aspects.

Besides the multiple atom mappings for one reaction, a similar issue exists when considering the atom mappings of several reactions. The Supplementary Figure S4 shows a case of two reactions involving two compounds with different atom mappings between them. In that case, the atom mapping of one reaction is not a superset of the atom mapping of the other reaction. This possibility makes it necessary to keep several reactions between some compounds, that is, multiple arcs are a necessity in general.

2.2 Finding multiple optimal routes

When using RouteSearch, the user specifies the numerical values of two parameters to evaluate the use of reactions in a route. These values are called *reaction costs*. One, denoted $C.r_n$, is for using any one reaction from the native database in a path. The other cost, denoted $C.r_l$, is for using any one reaction from the library. Note that by specifying costs such that $C.r_n < C.r_l$, which is typical, the optimal (minimal cost) routes have a tendency to select the reactions from the native database (e.g. EcoCyc) instead of the library (e.g. MetaCyc).

Besides the two costs just mentioned, a third cost for atom mapping is also specified by the user. That *atom mapping cost*, denoted $C.a$, is the cost of losing one atom from the source to the target compound along a metabolic route.

The atom species (e.g. carbon, oxygen) tracked during the computations is given by the set parameter S.

A valid route, or simply a route, from a start compound s to a target compound t is a sequence of reactions $R = \{r_1, \ldots, r_n\}$ where reaction r_1 uses compound s, reaction r_n produces compound t and reaction r_i produces a compound used by reaction r_{i+1}, $1 \le i < n$. Let R_n and R_l be the set of native and library reactions, respectively, of R. The cost of route R is

$$|R_n|C.r_n + |R_l|C.r_l + kC.a \tag{1}$$

where $|R_n|$ is the cardinality of R_n, $|R_l|$ is the cardinality of R_l and k is the number of atoms, having atom species in set S, of start compound s that did not reach the target compound t, according to the atom mappings of the reactions of R.

A best route R is such that no other route has a lower cost. Notice that several best routes may exist because several routes may have the same lowest cost.

The general problem of finding the minimum-cost routes in a network has been extensively studied resulting in various algorithms (Eppstein, 1998). Network algorithms such as the least-cost route searches of Bellman–Ford (Bellman, 1958; Ford, 1956; Hu and Shing, 1982) assume that the arc costs are fixed. This assumption does not hold when atom tracing is used as part of the evaluation criteria of a route. When considering the cost of a reaction to use between two vertices, the number of atoms lost, because the atom mapping of the reaction between these two vertices, is part of its cost. The atoms mapped are not static and depend on the route taken so far. Therefore, the algorithms described by Eppstein (1998) cannot be used to find the best route as defined above.

In this work, we use a Branch-and-Bound algorithm, which in general is computationally more demanding but where heuristics to prune the search can be applied.

Atom mappings and multiple arcs bring a difficulty that applies to many search algorithms such as the k-shortest route algorithm developed by Eppstein (1998) and used in MetaRoute (Blum and Kohlbacher, 2008a, b). Some of this difficulty was presented in Section 2.1 because multiple atom mappings for one reaction are handled during the construction of the mapped network.

Figure 1 shows the Find-Multiple-Optimal-Routes algorithm for finding optimal routes that rely on the recursive function Branch-and-Bound. In general, a Branch-and-Bound search estimates an upper bound on the minimum value of the optimal route. This is the *Bound* part in Branch-and-Bound. It is an essential element of the search, as it is used to prune the large space of possible routes.

A set of side compounds, intermediary compounds and reactions to avoid in any route can be given to the searching algorithm. But these sets are actually used during the initialization of the mapped network before the search: the arcs representing reactions to avoid, or reactions that use any of the side compound to avoid, are marked as inactive; likewise, the compounds (vertices) to avoid are marked as inactive. These inactive arcs and vertices are not used during the search. The initialization of the mapped network is not shown in Figure 1. These sets cannot be provided by using the current version (17.5 version) of the graphical user interface of RouteSearch but will be available in the next version.

During the search, we keep track of at most n best routes found so far. When n routes have been found, not necessarily the top n best routes, the route with the highest cost provides a *global upper bound* denoted *ub*. The bound *ub* is used to prune the search: when the value of a partial route has a value higher than *ub*, that partial route, and anything extending it, can be abandoned. This bound exists only after finding at least n routes to the target compound. The lower the upper bound *ub* is, the larger the number of routes pruned. Without pruning, the search would go over all possible routes from the source to the target, and potentially hundred of thousands of routes exist, if not more. Therefore, pruning as much as possible is important to make this algorithm practical. This pruning saves computation time but does not affect the optimality of the routes found.

Pruning mostly occurs on partial routes. The decision to prune a partial route is based on its lower bound value *lb*, which is the lowest possible cost to reach the target. To compute the lower bound *lb* in function Expand-Route, every vertex in the mapped

```
// Find at most $n > 1$ minimum cost metabolic routes in mapped
// network $N(V, A)$ from compound $s$ to compound $t$ with at
// most $l > 0$ reactions tracing atom species $S$ based on costs $C$.
Find-Multiple-Optimal-Routes($V, A, s, t, n, l, C, S$)
  Compute-Minimum-Costs($V, A, t$)
  // The source compound is the root of the search.
  $s.c \leftarrow 0$  // Cost is zero at root.
  $s.p \leftarrow \perp$ // Root of route has no predecessors.
  $s.M \leftarrow$ list of atoms of $s$ restricted to species of $S$
  // Search optimal routes starting at $s$.
  Branch-and-Bound($V, A, s, t, n, l, C, S, \infty$)
End Find-Multiple-Optimal-Routes

// Recursively search for $n$ minimum cost routes of at most
// $l$ reactions starting at node $v$ and ending at $t$.
Branch-and-Bound($V, A, v, t, n, l, C, S, ub$)
  For each $(r, c, u, M)$ in Expand-Route($V, A, v, t, d, C, ub$) Do
    If $u = t$ Then
      A new route to $t$ of cost $c$ has been found and can be
      constructed backward from $t$ to $s$ by using the $p$ fields.
      Keep this route if no more than $n$ routes have been kept or
      if $c$ is less than the costs of the $n$ routes found so far.
      If $n$ routes have been found, set the upper bound $ub$ to
      the largest cost of these routes.
    Else
      // Add vertex $u$ to current route ending at $v$ and search
      // further from $u$ if route can have more than one reaction.
      $u.p \leftarrow (r, v)$  // $v$ is the predecessor of $u$ via $r$ in route.
      $u.M \leftarrow M$     // The atoms traced at compound $u$.
      $u.c \leftarrow c$     // The cost at that compound $u$.
      If $l \geq 2$ Then
        Branch-and-Bound($V, A, u, t, n, l-1, C, S, ub$)
End Branch-and-Bound

// Returns a list of reactions to expand the route from vertex $v$.
// Apply a lower bound when possible to prune the search.
Expand-Route($V, A, v, t, d, C, ub$)
  For each $(v, u, R) \in A$ Do
    For each $(r, M) \in R$ Do
      $(c, M') \leftarrow$ Evaluate-Cost-Reaction($r, u.M, M, C$)
    $lb \leftarrow v.c + c + u.c_{min}$
    If $lb < ub$ Then
      Add $(r, lb, c, u, M')$ to list $Rs$
  Return sorted $Rs$ in increasing order of the lower bounds $lb$.
End Expand-Route

Evaluate-Cost-Reaction($r, aM, rM, C$)
  Let $c$ be the cost $C.r$ of the reaction plus the sum of the
  costs $C.a$ of the atoms lost by the mapping $rM$ given the
  atoms $aM$ at a vertex.
  Let $M'$ be the set of atoms not lost.
  Return $(c, M')$
End Evaluate-Cost-Reaction
```

Fig. 1. The algorithm to find multiple minimum cost metabolic routes, from a source to a target compound in a mapped network based on a Branch-and-Bound search. The routes are bounded in length by parameter *l*, and the costs *C* are selected by the user. See Equation (1) for the cost of a route

network has a minimum cost to the target compound, denoted $u.c_{min}$ for vertex *u*. The lower bound is the real cost to reach *u* that accounts for the atom mappings up to *u*, plus the minimum cost $u.c_{min}$.

The value $u.c_{min}$ is solely based on the shortest path and the cost of reactions to reach the target without taking into account any atom mappings. In other words, the values $u.c_{min}$ are strict lower bounds for the real cost of the route from *u* to the target. These minima are computed by the algorithm shown in Figure 2. This algorithm has a worst time complexity linear on the number of compounds and arcs in the network [that is, its running time is in $O(n+m)$ where *n* is the number of vertices (i.e. compounds), and *m* is the number of arcs]. This algorithm is fast in practice and typically takes less than a millisecond using a workstation with a CPU of 2.3 GHz on mapped networks with ~14 000 vertices. Notice that these minima are computed by starting from the target compound and going backward to all compounds in the network.

A Branch-and-Bound search can use one of several strategies when selecting the next vertex to expand a route. This selection is the *branch* part of Branch-and-Bound. In particular, it could select a vertex among all previous vertices visited so far or restrict the selection to the successor vertices of the currently visited vertex (i.e. the end-vertex of the current partial route). We use the latter strategy, which is called a depth-first search. Its advantage is a reduced amount of memory usage. Its disadvantage is a greater probability of doing more computation on suboptimal routes.

When extending a partial route in function Expand-Route, a heuristic is used to order the selection of the next compound from a list of possible compounds. That heuristic should be such that it helps find low cost routes first, which typically decreases the time to complete the search. The heuristic used is based on the lower bounds *lb* of the partial routes. This heuristic *does not* affect the optimality of the routes found, but helps increase the speed of the algorithm.

The efficiency of Find-Multiple-Optimal-Routes highly depends on the number of routes to find and their maximum length. To increase speed, a maximum route length and a maximum number of routes to find need to be provided as input, and ultimately by the user. Table 1 presents some examples of the time to execute this algorithm. All cases presented in Table 1 are a few seconds or less, but increasing the maximum length or the number of routes to find would eventually increase substantially the execution time.

For RouteSearch, a time limit to search for routes can be specified by the user. If the time limit is reached, several routes can be found before that. These routes are considered suboptimal because routes with lower costs could have been found.

3 IMPLEMENTATION

We give a brief description of the graphical user interface of RouteSearch that is accessible using a web browser such as Firefox or Chrome. Some browsers (Internet Explorer) cannot be used because they have limited capability for Scalable Vector Graphics, which is used to display the routes found.

The interface (see Supplementary Fig. S5) consists of an entry form that allows setting various parameters, such as the costs.

After entering a start and goal metabolite, clicking on the ‘Search Route’ button will start RouteSearch. The found routes are displayed in the region beneath the entry form. The routes are sorted in ascending order of their cost (best routes are presented first). If the search timed out, a corresponding message is shown.

Routes are displayed as a series of chemical structures for the metabolites within the path, connected by reaction arrows. The conserved moieties within each metabolite are shown using colors. Mousing over an atom of a compound highlights that atom in all compounds that contain it so that the user can easily identify each atom conserved or lost along a route. If a reaction is from MetaCyc, its arrow is red; if it is from the native database, its arrow is gray. Moving the mouse cursor over the arrow will pop up a tooltip describing the reaction in more detail, and clicking on the arrow opens up a new browser page displaying a description of the reaction with its complete atom mapping.

```
// Compute the minimum costs $u.c_{min}$ to reach the target compound
// $t$ from all compounds $u$ in the mapped network $N(V, A)$.
Compute-Minimum-Costs($V, A, t$)
  For each $v \in V$ Do $v.c_{min} \leftarrow \infty$
  $t.c_{min} \leftarrow 0$
  $Q \leftarrow \{t\}$        // $Q$ is a queue of vertices to process.
  Repeat until $Q = \emptyset$
    $v \leftarrow$ pop($Q$)
    // Set the cost $c_{min}$ of the unvisited predecessors $P$ of $v$.
    For each $u \in v.P$ Do
      If $u.c_{min} = \infty$ Then
        $c_{min} \leftarrow$ minimum cost reaction from $u$ to $v$
        $u.c_{min} \leftarrow v.c_{min} + c_{min}$
        add $u$ to queue $Q$
End Compute-Minimum-Costs
```

Fig. 2. The algorithm to compute the minimum costs from all compounds to a target compound before applying the Branch-and-Bound algorithm. These costs are solely based on the constant costs of reactions and are used as a heuristic to evaluate the future cost of a partial route. On a 10 000 compounds network, using a 2.3 GHz computer, the implementation of this algorithm takes less than a millisecond to run. Note that the values $u.c_{min}$ are computed backward from the target compound t to every compound u

4 DISCUSSION

4.1 Empirical validation of RouteSearch

To validate RouteSearch against published pathways that were engineered into *Escherichia coli*, we ran RouteSearch on the examples summarized in Table 1. The common setup for the examples was to select EcoCyc as the native database and to allow using MetaCyc as the library. For all searches, we used the parameter defaults: a cost of 5 for native reactions, a cost of 10 for reactions coming from the library and a cost of 15 for any atom lost. We selected these default values to favor the use of the native reactions over the reactions of the library, and avoid losing atoms in a route if adding reactions from the library would do so. The maximum route length was 9, and the requested number of routes was 10.

These examples were done on a network that includes EcoCyc and MetaCyc (both version 17.5), which has 14 005 vertices (i.e. compounds) and 22 473 arcs. We use a prerelease 18.0 version of Pathway Tools. A 2.3 GHz computer requires ~5 s of CPU time to create this combined network. These times were not included in the search speed reported in the following sections. Most of the time is spent preparing the restricted atom mappings between specific compounds. Once such networks are created, they can be reused for any number of searches.

4.1.1 From L-tyrosine to umbelliferone The optimal route found by RouteSearch corresponds to the engineered pathway described in Santos *et al.* (2011), Vannelli *et al.* (2007) and Vialart *et al.* (2012). All of its four reactions came from the library, and the total cost was 70.

Table 1. Searches done using RouteSearch with the optimal routes found and the time taken by the Find-Multiple-Optimal-Routes algorithm

Source to target	Number of reactions	Atoms conserved	Route as main compounds	Time in seconds
L-tyrosine to umbelliferone	4	9 carbons 2 oxygens	L-tyrosine, 4-coumarate, 4-coumaryl-CoA, 2,4-dihydroxycinnamoyl-CoA, umbelliferone	3.6
Pyruvate to isobutanol	5	2 carbons	Pyruvate, (S)-2-acetolactate, 2,3-dihydroxy-3-methylbutanoate, 3-methyl-2-oxobutanoate, isobutanal, isobutanol	2.2
3-methyl-2-oxobutanoate to 3-methylbutanol	7	4 carbons	3-methyl-2-oxobutanoate, (2S)-2-isopropylmalate, 2-isopropylmaleate, (2R,3S)-3-isopropylmalate, (2S)-2-isopropyl-3-oxosuccinate, 4-methyl-2-oxopentanoate, 3-methylbutanal, 3-methylbutanol	0.9
Aldehydo-D-xylose to ethylene glycol	4	2 carbons 2 oxygens	Aldehydo-D-xylose, D-xylonate, 2-dehydro-3-deoxy-D-arabinonate, glycolaldehyde, ethylene glycol	0.3
Beta-D-glucose to 1,3-propanediol	6	3 carbons 2 oxygens	Beta-D-glucose, beta-D-glucose 6-phosphate, Beta-D-fructofuranose 6-phosphate, dihydroxyacetone, glycerol, 3-hydroxypropionaldehyde, 1,3-propanediol	0.9

Note: All searches used a native reaction cost ($C.r_n$) of 5; a library reaction cost ($C.r_l$) of 10; a lost atom cost ($C.a$) of 15; a maximum path length of 9; and a maximum of 10 routes to find. The network searched contains reactions from EcoCyc and MetaCyc.

4.1.2 From pyruvate to isobutanol The third ranked route found by RouteSearch corresponds to the engineered pathway described in Atsumi *et al.* (2008, 2010) and Savrasova et al. (2011). Its last two reactions came from the library, and the total cost was 95. In this route, only two atoms were conserved from source to target. The reason was that, for the first reaction (acetolactate synthase), only one atom mapping existed, compared with the recomputed atom mappings in a later version of EcoCyc, where two atom mappings existed, one of which retains more atoms. In the later EcoCyc version, this route became the top optimal route with a cost of 80, and 3 atoms kept.

4.1.3 From 3-methyl-2-oxobutanoate to 3-methylbutanol The fourth ranked route corresponds to the engineered pathway described in Atsumi *et al.* (2008) and Connor and Liao (2008). The last two reactions came from the library, and the total cost was 105. One of the two top-ranked routes found by RouteSearch is essentially equivalent to the fourth ranked route, except shorter by one reaction. It is shorter because MetaCyc contains a single reaction that is equivalent to what is represented by two subreactions in EcoCyc. The other top-ranked route is even shorter by another reaction, for the same reason. A single reaction in the library is biochemically equivalent to two subreactions.

4.1.4 From Aldehydo-D-Xylose to ethylene glycol The second ranked route corresponds to the engineered pathway described in Liu et al. (2013). All reactions came from the library, and the total cost was 130. The first (optimal) route found by RouteSearch differs in the last step of the predicted route, which uses the cofactor NADH instead of the reaction that uses NADPH, as was described in the literature. However, the reaction using NADPH is not present in EcoCyc. Thus, importing it from the library increases the cost of finding the published route. The NADPH reaction was chosen to achieve better redox balance inside the engineered cell.

4.1.5 From beta-D-glucose to 1,3-propanediol The engineered pathway producing 1,3-propanediol from beta-D-glucose of Nakamura and Whited (2003) illustrates how removing a reaction from consideration by RouteSearch can affect the search results.

Interestingly, although the optimal route found by RouteSearch was similar in spirit to the engineered pathway, it was shorter by two reactions and appears to be a more efficient route overall. The key difference is that fructose-1,6-bisphosphate is not used as an intermediate, and one ATP can be saved. Instead of phosphorylating beta-D-fructofuranose 6-phosphate, the reaction (RXN0-313) for the EcoCyc enzymes fructose 6-phosphate aldolase (FsaA/FsaB) is used to generate the unphosphorylated intermediate dihydroxyacetone, which is then reduced to glycerol directly. The enzymes FsaA/FsaB have been recently discovered, representing a novel activity first found in *E. coli* (Schurmann and Sprenger, 2001). They have not been well studied, so it is not yet clear whether this shortcut would be viable for high-volume production. This route contains six reactions and carries a cost of 140.

To find the route for the published pathway, two additional reactions from MetaCyc would have to replace the shortcut, thus raising the cost of the route to 160, because the cost of importing one reaction from the library was set at 10.

After the requested number of routes was set to 60, all routes with a cost of 160 were included in the results (routes 19–54). Among these results, routes 48 and 49 corresponded most closely to the published engineered pathway, passing through fructose-1,6-bisphosphate and necessitating eight reactions. However, one step, going from sn-glycerol 3-phosphate to glycerol, did not use the reaction in the engineered pathway, but instead used either of two reactions that have different side compounds. The reason was that the existing MetaCyc reaction for EC 3.1.3.21 used the non-stereospecific compound glycerol 1-phosphate, which did not connect with the intermediates used in the routes. In a later version of MetaCyc, a stereospecific variant of the reaction was added, leading to inclusion of the published route in the results.

Rerunning the example using reaction RXN0-313 disallowed now caused the route with the shortcut to disappear, and the routes corresponding to the published engineered pathway moved closer to the top.

4.2 Related work

For the software tools listed in this section, we stress whether reaction atom mapping (i.e. atom tracking) is used, and if so, whether the number of atoms lost in a route is part of their optimality criteria. We also categorize them into three groups according to the classification presented by Kotera *et al.* (2013): (i) using only predefined networks, as in RouteSearch, to find routes, (ii) inferring missing compounds, with corresponding reactions, to find new routes and (iii) inferring missing reactions to find new routes.

Heath *et al.* (2010) present a metabolic route searching algorithm, classified in group A, using the atom mappings of the KEGG RPAIR database. The route found is generally a *branched pathway* containing multiple linear routes. The optimality criterion is not based on the atoms lost in a route. Instead, the atom mappings are used to discard routes that do not keep a minimum of atoms. They use a network of 5844 compounds and 7340 reactions that is substantially smaller than the EcoCyc/MetaCyc network that we use in our work. They report timing results for three searches, 8 min from chorismate to (S)-norcoclaurine, 24 min from sn-glycero-3-phosphocholine to L-threonine, and 30 min from α-D-glucose 6-phosphate to L-tryptophan. These timing results are much higher than the timing results of RouteSearch. For example, RouteSearch finds in EcoCyc an optimal linear route from α-D-glucose 6-phosphate to L-tryptophan in <4 s. Their algorithm was implemented in Java and it is available as an online tool at the Kavrakilab Web site.

PathPred (Moriya *et al.*, 2010), classified in group B, focuses on discovering new metabolic routes between two compounds using the KEGG RPAIR atom-mapping database where the target or source compound may not exist in the database. The metabolic routes are restricted to microbial biodegradation of environmental compounds and biosynthesis of plant secondary metabolites. The algorithm does not find optimal routes based on atom mappings.

ReTrace (Pitkänen *et al.*, 2009), classified in group A, is a metabolic route-finding software. It takes into account atom mappings of reactions, and finds branched pathways. Several source compounds can be specified. The genome-scale network used is based on the KEGG databases, including the RPAIR atom-mapping database. The algorithm used does not guarantee optimality of the solution found. The approach is graph-based as in RouteSearch, but the vertices of the graph are atoms, not compounds. The computation times reported in the article are high. On a cluster of 25 computers, with Intel Pentium 2.8 GHz CPUs, finding 10 optimal routes from one source compound to a target compound is on the order of thousands of seconds.

MetaRoute (Blum and Kohlbacher, 2008a), classified in group A, is an interactive web-based metabolic route searching tool similar to RouteSearch. It also accounts for the atom mappings of reactions. The atom-mapping rules used in MetaRoute are described in Blum and Kohlbacher (2008b) and are based on the KEGG database of reactions. The Eppstein's k-shortest route algorithm (Eppstein, 1998) is used for searching multiple optimal routes. The optimality criterion is not based on the atoms lost in a path but instead on the connectivity of the vertices in the network. The atom mappings are used to avoid searching routes that lose all atoms. This optimality criterion was chosen with the objective of finding textbook pathways. This is different from RouteSearch, where the number of atoms lost in a path is of primary importance. The paper (Blum and Kohlbacher, 2008a) does not report any computation speed, although the interactive web tools report that the search can take between 10 and 70 s, depending on the size of the reaction networks used. From the Web site, MetaRoute found one optimal route from α-D-glucose to fumarate in *E. coli* K-12 MG1655 in about 10 s. The route found had 13 reactions and transfers 3 carbon atoms to fumarate. On the other hand, RouteSearch found a shorter route of 10 reactions transferring the same number of carbon atoms and took about 2 s to find. In MetaRoute, searching for routes tracing several user-specified atom species at the same time (e.g. oxygen and carbon) is not possible, but a user can specify to trace any combination of oxygen, carbon, nitrogen, sulfur and phosphate with RouteSearch. With MetaRoute, suggesting a cost for the atoms lost is not possible, nor the cost of the reactions coming from a library of reactions. This is a major shortcoming for metabolic engineering because maximizing the conservation of atoms along routes is not possible, nor is minimizing the number of new reactions added to an organism.

Pathway Hunter (Rahman *et al.*, 2005) finds the shortest paths between two compounds in a reaction network based on the similarity of compounds. Croes *et al.* (2005) search for the shortest routes in genome-scale reaction network. PathMiner (McShan *et al.*, 2003) can search for optimal metabolic routes in genome-scale reaction networks using an A^* algorithm. The optimality criterion is the minimum sum of dissimilar compounds along a route plus the number of reactions. These three tools, classified in group A, do not account for atom mappings and therefore do not base their optimal routes on the atoms lost, an important element of the optimality criteria of RouteSearch.

5 CONCLUSION

We have presented an algorithm to efficiently compute the optimal metabolic routes in genome-scale reaction networks, accounting for atom loses when evaluating routes. This algorithm is used in RouteSearch, a publicly accessible web-based tool at BioCyc.org.

The Branch-and-Bound-based algorithm Find-Multiple-Optimal-Routes can find several optimal routes from a source to a target compound. This algorithm's efficiency depends on the size of the network, the maximum length of the route to find and the number of routes to find. The algorithm was validated on several examples from the metabolic-engineering literature on large metabolic networks (i.e. EcoCyc combined with MetaCyc has ∼14 000 vertices and 22 000 arcs). The optimal route computed by the algorithm matched the published solution in one case; for the remaining four problems, RouteSearch found the published solution as one of the best-scored solutions. Most of these problems were solved in less than 5 s, which is significantly faster than previous systems.

As far as we know, RouteSearch is the first published exact algorithm guaranteeing optimality of the routes found when the cost of the atoms lost in the routes are part of the optimality criteria. Previous works use atom mappings in some way, like avoiding routes that do not keep at least one atom from the source to the target compound, but not as part of the optimality criteria to optimize.

Future work should consider the symmetry of compounds for atom mappings so that the atoms lost and conserved in routes are more precisely tracked. Practically, not having to specify a maximum length or no a priori maximum number of routes to find would be useful. The algorithm should also be extended to consider the cost of side compounds, the toxicity of some of the metabolites produced and the taxonomic range of the new enzymes introduced in routes. Finding optimal routes from multiple start compounds and to multiple target compounds is more complex because the possible candidate solutions are no longer linear routes but multiple linear routes merged together. A Branch-and-Bound approach could still be used but the search space would be different.

Funding: Office of Science (BER), U.S. Department of Energy under Award Number(s) (DE-SC0004878). The content of this article is solely the responsibility of the authors and does not necessarily represent the official views of the Department of Energy.

Conflict of Interest: none declared.

REFERENCES

Atsumi,S. *et al.* (2008) Non-fermentative pathways for synthesis of branched-chain higher alcohols as biofuels. *Nature*, **451**, 86–89.

Atsumi,S. *et al.* (2010) Engineering the isobutanol biosynthetic pathway in *Escherichia coli* by comparison of three aldehyde reductase/alcohol dehydrogenase genes. *Appl. Microbiol. Biotechnol.*, **85**, 651–657.

Bellman,R. (1958) On a routing problem. *Q. App. Math.*, **16**, 87–90.

Blum,T. and Kohlbacher,O. (2008a) MetaRoute: fast search for relevant metabolic routes for interactive network navigation and visualization. *Bioinformatics*, **24**, 2108–2109.

Blum,T. and Kohlbacher,O. (2008b) Using atom mapping rules for an improved detection of relevant routes in weighted metabolic networks. *J. Comput. Biol.*, **15**, 565–576.

Caspi,R. *et al.* (2012) The MetaCyc database of metabolic pathways and enzymes and the BioCyc collection of pathway/genome databases. *Nucleic Acids Res.*, **40**, D742–D753.

Connor,M.R. and Liao,J.C. (2008) Engineering of an *Escherichia coli* strain for the production of 3-methyl-1-butanol. *Appl. Environ. Microbiol.*, **74**, 5769–5775.

Croes,D. *et al.* (2005) Metabolic PathFinding: inferring relevant pathways in biochemical networks. *Nucleic Acids Res.*, **33** (**Suppl. 2**), W326–W330.

Eppstein,D. (1998) Finding the *k* shortest paths. *SIAM J. Comput.*, **28**, 652–673.

Ford,L. (1956) *Network Flow Theory*. Rand Corporation. Santa Monica, CA, USA.

Heath,A.P. *et al.* (2010) Finding metabolic pathways using atom tracking. *Bioinformatics*, **26**, 1548–1555.

Hu,T.C. and Shing,M.T. (1982) *Combinatorial Algorithms*. Dover Publications, Mineola, N.Y.

Kanehisa,M. *et al.* (2012) KEGG for integration and interpretation of large-scale molecular data sets. *Nucleic Acids Res.*, **40**, D109–D114.

Karp,P. *et al.* (2010) Pathway Tools version 13.0: integrated software for pathway/genome informatics and systems biology. *Brief. Bioinform.*, **11**, 40–79.

Kotera,M. *et al.* (2013) Supervised de novo reconstruction of metabolic pathways from metabolome-scale compound sets. *Bioinformatics*, **29**, i135–i144.

Latendresse,M. *et al.* (2012) Accurate atom-mapping computation for biochemical reactions. *J. Chem. Inf. Model.*, **52**, 2970–2982.

Liu,H. *et al.* (2013) Biosynthesis of ethylene glycol in *Escherichia coli*. *Appl. Microbiol. Biotechnol.*, **97**, 3409–3417.

McShan,D. *et al.* (2003) PathMiner: predicting metabolic pathways by heuristic search. *Bioinformatics*, **19**, 1692–1698.

Moriya,Y. *et al.* (2010) PathPred: an enzyme-catalyzed metabolic pathway prediction server. *Nucleic Acids Res.*, **38**, W138–W143.

Nakamura,C.E. and Whited,G.M. (2003) Metabolic engineering for the microbial production of 1,3-propanediol. *Curr. Opin. Biotechnol.*, **14**, 454–459.

Pitkänen,E. *et al.* (2009) Inferring branching pathways in genome-scale metabolic networks. *BMC Syst. Biol.*, **3**, 1–22.

Rahman,S.A. *et al.* (2005) Metabolic pathway analysis web service (Pathway Hunter tool at cubic). *Bioinformatics*, **21**, 1189–1193.

Santos,C.N. *et al.* (2011) Optimization of a heterologous pathway for the production of flavonoids from glucose. *Metab. Eng.*, **13**, 392–400.

Savrasova,E.A. *et al.* (2011) Use of the valine biosynthetic pathway to convert glucose into isobutanol. *J. Ind. Microbiol. Biotechnol.*, **38**, 1287–1294.

Schurmann,M. and Sprenger,G.A. (2001) Fructose-6-phosphate aldolase is a novel class i aldolase from *Escherichia coli* and is related to a novel group of bacterial transaldolases. *J. Biol. Chem.*, **276**, 11055–11061.

Vannelli,T. *et al.* (2007) Production of p-hydroxycinnamic acid from glucose in *Saccharomyces cerevisiae* and *Escherichia coli* by expression of heterologous genes from plants and fungi. *Metab. Eng.*, **9**, 142–151.

Vialart,G. *et al.* (2012) A 2-oxoglutarate-dependent dioxygenase from *Ruta graveolens L.* exhibits p-coumaroyl CoA 2′-hydroxylase activity (C2'H): a missing step in the synthesis of umbelliferone in plants. *Plant J.*, **70**, 460–470.