HAS-SOP: ГИБРИДНАЯ МУРАВЬИНАЯ СИСТЕМА ДЛЯ ЗАДАЧИ ПОСЛЕДОВАТЕЛЬНОГО УПОРЯДОЧЕНИЯ

Luca Maria Gambardella, Marco Dorigo

Аннотация

Мы представляем HAS-SOP, новый подход к задаче последовательного упорядочения. HAS-SOP сочетает в себе муравьиный алгоритм, метаэвристику, основанную на популяции, с новым локальным оптимизатором, расширение TSP-эвристики, которая непосредственно работает с несколькими ограничениями без увеличения вычислительной сложности. Мы провели сравнение различных реализаций HAS-SOP и представляем новую структуру данных, которая повышает производительность системы. Экспериментальные результаты на наборе из двадцати трех тестовых задач, взятые из TSPLIB, показывают, что HAS-SOP превосходит существующие методы как с точки зрения качества решения, так и вычислительного времени. Кроме того, HAS-SOP улучшает большинство наиболее известных результатов для рассматриваемой задачи.

Введение

Существует много NP-сложных задач комбинаторной оптимизации, для которых нецелесообразно находить оптимальное решение. Одной из них является задача последовательного упорядочения (SOP). Для таких задач разумно только искать эвристические алгоритмы, которые быстро находят хорошие, хотя и не обязательно оптимальные, решения. Эти алгоритмы часто используют некоторые знания специфических задач либо для построения, либо для улучшения решений. В последнее время многие исследователи сосредоточили свое внимание на новом классе метаэвристических алгоритмов. Метаэвристики – довольно общие алгоритмические структуры, которые могут быть применены к нескольким различным задачам оптимизации с несколькими изменениями. Примерами метаэвристики являются имитация отжига, эволюционные вычисления и табу-поиск. Метаэвристики зачастую были вдохновляемы природными процессами. В самом деле, приведенные выше метаэвристики были вдохновлены соответственно физическим процессом отжига, дарвиновским эволюционным процессом и умным управлением структурами памяти. Одним из самых последних природо-вдохновленных метаэвристик является муравьиный алгоритм (Dorigo, Maniezzo and Colorni, 1991, 1996; Dorigo, 1992). Так, вдохновляющий естественный процесс - это поведение питания муравьев. Муравьиная система (ACS), частный случай муравьиного алгоритма, была недавно показана (Dorigo and Gambardella, 1997) на симметричных и асимметричных задачах коммивояжера для конкурирования с другими метаэвриситками. Хотя это интересный и обнадеживающий результат, по-прежнему очевидно, что муравьиный алгоритм, а также другие метаэвристики, во многих случаях не могут конкурировать со специализированными методами локальными поиска. Поэтому нынешняя тенденция (Johnson and McGeoch, 1997) – связать с метаэвритиками локальный оптимизатор. Это интересное сочетание, так как локальный оптимизатор часто страдает от проблемы «инициализации». Таким образом, выполнение локальной оптимизации – часто функция типа начального решения, к которому она применена. Например, многозаходных, то есть, применением локального поиска для различных начальных решений, было установлено, что это плохой выбор, так как локальный поиск тратит большую часть своего времени на повышение начального решения низкого качества (Aarts and Lenstra, 1997). Поэтому становится интересно найти хорошие метаэвристические локальные оптимальные сцепления, где сцепление хорошо, если метаэвристика генерирует начальные решения, которые можно отнести к очень хорошим локальным оптимумам локальным оптимизатором.

В предыдущей работе (Dorigo and Gambardella, 1997) нами было показано, что связью ACS с расширенной версией процедуры 3-opt локального поиска можно получать высококачественные решения, как для симметричных, так и для асимметричных задач

коммивояжера (TSP). Кроме того, мы недавно применяли HAS-SOP, муравьиный алгоритм в сочетании с простой формой локального поиска к квадратичной задачи о назначениях (QAP). HAS-SOP произвел лучшие решения (Gambardella, Taillard and Dorigo, 1997), чем лучшие известные алгоритмы по экземплярам задач реального мира (сравнение включает реактивный табу-поиск (Battiti and Tecchioli, 1994), надежный табу-поиск (Taillard, 1991), имитации отжига (Коннолли, 1990) и генетический гибридный поиск (Fleurent and Ферланд, 1994)).

В этой статье мы атакуем задачу последовательного упорядочения (SOP) с помощью муравьиного алгоритма в сочетании с модифицированной версии процедурой 3-opt поиска. В результате гибридная муравьиная система для SOP (HAS-SOP) превосходит все известные эвристические подходы к SOP. Кроме того, нам удалось усовершенствовать многие из лучших результатов, опубликованных в TSPLIB, одним из наиболее важных баз данных сложных TSP-связанных оптимизационных задач, доступных по Интернету.

1.1 Задача последовательного упорядочения

Задача последовательного упорядочения с ограничениями предшествования (SOP) впервые была сформулирована Эскудеро (1988), чтобы проектировать эвристику для системы планирования производства. Она состоит из нахождения минимального взвешенного гамильтонового пути на направленном графе с весами на дугах и узлах, подчиненных ограничениям предшествования среди узлов.

Полагаем, что полный граф G = (V, A), где узел V и дуга A, где узлы соответствуют рабочим местам 0...,*i*.., n (n+1 = |V|). Каждой дуге *(i,j)* соответствует стоимость tij∈ R с tij≥0, эта стоимость представляет собой требуемое время ожидания между концом работы *i* и начало работы *j*. Каждому узлу *i* соответствует стоимость p∈ R с pi>0, который представляет собой продолжительность обработки работы *j*. Набор узлов V включает в себя стартовый узел (узел 0) и заключительный узел (узел n) связанный со всеми другими узлами. Затраты между узлом 0 и другими узлами равны времени установки узла i, t0i=pi∀i, и tin=0 ∀ i.

Ограничения предшествования задаются дополнительным нециклическим диграфом P=(V, R), который определен на том же самом наборе узлов V. Дуга (ij)∈R, если работа i должна предшествовать работе j в каком-нибудь выполнимом решении R имеет транзитивное свойство (то есть, если (ij)∈R и (i,k)∈R тогда (i,k)∈R*)*. Так как последовательность всегда начинается в узле 0 и концах в узле n, 0,i∈R ∀i∈V\{0}, и i,n∈R ∀i∈V\{n}. Б будущем мы

укажем предшественником [i] и преемником [i] наборы узлов, которые должны предшествовать, следовать за узлом *i в* любом выполнимом решении.

Данные выше определения, задача SOP может быть определена как задача нахождения последовательности работы, подчиненной ограничениям предшествования. Поэтому эквивалентно задаче нахождения возможного гамильтонового пути с минимальной стоимостью в G при ограничениях предшествования, данных P.

SOP может также быть сформулирована как общий случай асимметричной задачи коммивояжера (ATSP), основываясь только на весах ребер (в SOP решение соединяет первый и последний город путем, который посещает все узлы один раз, в противоположность ATSP, в котором решение является закрытым туром, который посещает все узлы один раз). Эта формулировка эквивалентна предыдущему: достаточно удалить веса из узлов и пересмотреть вес c_{ij} ребра (i,j), добавляя к каждому t_{ij} вес p_j узла j. Б этом представлении c_{ij} – вес ребра (i,j) (где c_{ij} может отличаться от c_{ji}), который может или представить стоимость ребра (i,j), когда если $c_{ij} \geq 0$, или ограничение на порядок обхода, когда $c_{ij}=-1$ (значение $c_{ij}=-1$ означает, что узел i должен предшествовать узлу j, причем, необязательно последовательно). В этой статье мы будем использовать эту последнюю формулировку.

1.2 Эвристические методы для SOP

SOP моделируют такие задачи реального мира как планирование производства (Escudero, 1988), задачи маршрутизации одного транспортного средства с погрузкой и ограничениями поставки (Pulleyblank и Timlin, 1991; Savelsbergh, 1990), и транспортные задачи в гибких производственных системах (Ascheuer, 1996).

SOP может быть представлена как общий случай и асимметричного TSP, и задачи поставки и погрузки. Она отличается от ATSP, потому что первый и последний города зафиксированы, и в дополнительном наборе ограничений предшествования на порядок, в котором должны быть посещены города. Она отличается от задачи погрузки и поставки, потому что эта задача обычно основана на симметричном TSP, и потому что задача погрузки и поставки включает ряд ограничений между городами с уникальным предшественником, определенным для каждого города, в противовес SOP, где может быть определено многократное предшествование.

Подходы, основанные на ATSP

Задачи SOP были первоначально решены как ограниченные версии асимметричного TSP. Главное усилие было сделано в расширении математического определения задачи ATSP, вводом новых уравнений для моделирования ограничений. Первая математическая модель для задач SOP была введена в (Ascheuer и др., 1993), где был предложен подход секущих плоскостей, чтобы вычислить более низкие границы на оптимальном решении. В (Escudero и др., 1994) был описан метод Лагранжа relax-and-cut, и были определены новые действительные усечения для получения более сильных низких границ. Позже, основываясь на многогранном исследовании, выполненном на задачах ATSP с ограничениями предшествования (Balas, Fischetti и Pulleyblank, 1995), Ascheuer (1996) предложил новый класс действительных неравенств и описал алгоритм branch-and-cut для широкого класса случаев SOP. Его подход также исследует возможность вычислить и улучшить подоптимальные возможные решения задачи, начиная с верхней границы, вычисленной многогранным исследованием. Верхняя граница - начальное решение эвристической фазы, основанной на известных эвристиках ATSP, которые многократно применяются в целях улучшения допустимых решений. Эти эвристики не содержат ограничения напрямую: в результате невозможные решения просто отвергнуты. При таком подходе Ascheuer был в состоянии вычислить новые верхние границы для задач SOP в TSPLIB, хотя генетический алгоритм, называемый Maximum Partial Order/Arbitrary Insertion (MPO/AI), недавно предложенный Chen и Smith (1996), кажется, работает лучше в том же классе задач. MPO/AI всегда работает в пространстве допустимых решений путем введения сложных операторов скрещивания, что сохраняет общую схему из двух родителей путем определения их максимального частичного порядка через матричные операции. Новое решение закончено с использованием конструктивной эвристики.

Подходы, основанные на задаче погрузки и поставки

Эвристические подходы к задаче погрузки и поставки основаны на отдельных расширениях TSP эвристик, способных контролировать ограничения предшествования, улучшая выполнимые решения без какого-либо увеличения в вычислениях. Psaraftis (1983) ввел технику предварительной обработки для обеспечения возможной проверки в постоянное время, начиная алгоритм с процедуры сканирования, которая, по начальным затратам $O(n^2)$, получает матрицу выполнимости, которая содержит информацию о возможных обменах ребер. Впоследствии (Solomon, 1987) предложил процедуру поиска, основанную на подогнанном механизме обновления, в то время как (Savelsbergh, 1990; Van der Bruggen, Lenstra и Schuur, 1993; Kindervater и Savelsberg, 1997), представили лексикографическую стратегию поиска, вариацию традиционных edge-exchange TSP эвристик, которая уменьшает число посещаемых узлов, не теряя никакого возможного обмена. В целях обеспечения проверки ограничений в постоянном времени лексикографическая стратегия поиска была объединена с процедурой маркировки, где узлы в последовательности помечены информацией, связанной с их уникальным преемником/предшественником, и ряд глобальных переменных модернизирован, чтобы сохранять эту информацию действительной. Savelsbergh (1990) представил лексикографический поиск, основанный на 2-opt, и 3-opt стратегиях, который обменивает фиксированное число ребер, в то время как Van der Bruggen и др. (1993) предложили поиск переменной глубины, основанный на подходе Lin и Kernighan (1973).

К сожалению, эта процедура маркировки не применима в случае многократных ограничений предшествования, потому что она основана на наличии в последовательности узлов с уникальным преемником/предшественником. С другой стороны, лексикографическая стратегия поиска независима от числа ограничений предшествования и поэтому может использоваться для решения задач SOP, в которых позволяются многократные ограничения предшествования.

Подход к задаче SOP, представленный в этой статье, является первым в литературе, который использует расширение TSP эвристически, чтобы непосредственно контролировать многократные ограничения без какого-либо увеличения в вычислениях. Наш подход комбинирует конструктивную фазу, основанную на алгоритме ACS (Dorigo и Gambardella, 1997), с лексикографическим эвристическим поиском и с новой процедурой маркировки, способной контролировать многократные ограничения предшествования. Кроме того, мы проверяем и сравниваем различные методы выбора узлов в течение поиска и различных критериев остановки. В особенности мы проверяем две различных эвристики выбора: одна основана на don’t look bit структуре данных, введенная Bentley (1992), другая основана на новой структуре данных, называемой don’t push stack, введенная авторами.

2. Оптимизация муравьиной колонии

Оптимизация муравьиной колонии является подходом, основанный на популяции, в котором ряд агентов (искусственных муравьев) строят решения алгоритмом вероятностного ближайшего соседства, который использует специальную меру расстояния. Как только они построили решение, искусственные муравьи используют качество произведенных решений для обновления информации о расстоянии. Давайте рассмотрим, в целях представления, Эвклидову TSP. Каждое ребро графа имеет две связанных меры: близость η_{ij} и след феромона τ_{ij}. Близость, определенная как инверсия длины ребра, является статическим значением, то есть, она никогда не изменяется для приведенного примера задачи, в то время как след феромона – динамическое значение, изменяемое во времени муравьями. Каждый муравей имеет начальный город, и его цель состоит в том, чтобы построить решение, то есть, полный тур. Тур построен в каждом городе (города – вершины графа): когда муравей k находится в городе i, он выбирает для перемещения город j, используя вероятностное правило, которое одобряет города, которые являются близкими и связанными ребрами с высоким значением следа феромона. Города всегда выбираются среди еще не посещенных, чтобы обеспечить строительство выполнимых решений. Как только все муравьи построили полный тур, след феромона обновляется на посещенных ребрах. Руководящий принцип - увеличение следа феромона на ребрах, посещенных теми муравьями, которые находят короткий тур. След

феромона также испаряется так, чтобы память о прошлом была постепенно потеряна (это предотвращает плохие начальные решения, которые будут от наличия длительного эффекта на процесс поиска). Вышеупомянутый неофициально описанный алгоритм (см. также иллюстрацию 1) может быть осуществлен многими различными способами, и детали о конкретных решениях внедрения для TSP могут быть найдены в (Dorigo, Maniezzo и Colorni, 1991; 1996; Dorigo, 1992; Dorigo и Gambardella, 1997).

Понятно, что оптимизация муравьиной колонии может быть легко приспособлена к SOP: достаточно иметь муравьев, строящих путь от источника к пункту назначения, учитывая ограничения порядка обхода (это может легко быть достигнуто при наличии муравьев, выбирающих еще не посещенные города, которые не нарушают порядок предшествования).

1) Initialize the pheromone trail
2) For Imax iterations repeat:
2a) For each ant k = 1, ..., m, build a new
solution exploiting the pheromone trail
2b) Update the pheromone

Самый важный компонент оптимизации муравьиной колонии – управление следами феромона, которые используются в сочетании с целевой функцией для построения новых решений. Информация, содержащаяся в следах феромона, и использовании этой информации – ключевые элементы муравьиной системы. Неофициально уровни феромона дают меру того, как желательно вставить данный элемент в решение. Следы феромона используются для исследования и эксплуатации. Исследование касается вероятностного выбора компонентов, используемых для построения решения: более высокая вероятность дается элементам с сильным следом феромона. Эксплуатация выбирает компонент, который максимизирует смесь значений следа феромона и частичных оценок целевой функции.

3. Гибридная муравьиная система

Гибридная муравьиная система для SOP (HAS-SOP) – алгоритм муравьиной колонии, как описано выше, плюс локальный поиск. HAS-SOP строит ряд решений алгоритмом муравьиной колонии, и затем процедура локального поиска приводит их к локальному оптимуму. Эти локальные оптимальные решения используются для вычисления размера изменения в следах феромона. Следы феромона тогда обновляются, и HAS-SOP повторяется. Это продолжается, пока не встречено условие завершения. Условие завершения встречено, когда одно из следующих условий становится верным: было сгенерировано определенное число решений, истекло определенное время центрального процессора, или никакое усовершенствование не было достигнуто в течение последних NO_IMP итераций.

В дальнейшем мы дадим детальное описание и муравьиного алгоритма, и процедуры локального поиска.

3.1 Муравьиный алгоритм

Муравьиный алгоритм реализует конструктивную фазу HAS-SOP, цель которой – построить выполнимое решение SOP задачи. Он генерирует выполнимые решения с вычислительными затратами порядка O (n^2).

Неформально это работает следующим образом. Каждый муравей многократно начинает движение с узла 0 и добавляет новые узлы, пока все узлы не были посещены, и узел n достигнут. Находясь в узле i, муравей выбирает вероятностно следующий узел j из набора F (i) из выполнимых узлов. F (i) сдержит все узлы j, которые еще должны быть посещены, и все, что должны предшествовать j, согласно ограничениям предшествования, были уже вставлены в последовательность.

Система выбирает с вероятностью q_0 узел с лучшим $\tau_{ij} \cdot \eta_{ij}$, $j \in F(j)$ (детерминированное правило), в то время как с вероятностью $(1-q_0)$, узел выбран согласно вероятностному правилу, которое одобряет узлы, связанные ребрами с более высокими значениями

$$j = \begin{cases} \max_{j \in F(i)} (\tau_{ij} \cdot \eta_{ij}) \\ p_{ij}_{j \in F(i)} = \dfrac{\tau_{ij} \cdot \eta_{ij}}{\sum_{i \in F(i)} \tau_{ij} \cdot \eta_{ij}} \end{cases}$$, с вероятностью $(1-q_0)$

Значение q_0 выбирается из:

$\tau_{ij} \cdot \eta_{ij}, j \in F(i)$ $\qquad q_0 = 1 - \dfrac{S}{n}$ (1)

q_0 основан на параметре S, который является числом узлов. S позволяет системе определить q_0 независимо от задачи, измеряя его таким образом, что ожидаемое число узлов, отобранных с вероятностным правилом, является S.

Как только каждый муравей построил выполнимое решение, к нему применяют локальный поиск. В локальном масштабе оптимальные решения используются, чтобы обновить следы феромона на дугах, согласно правилу испарения следа феромона. В нашем случае только лучший муравей – муравей, который построил самый короткий тур, может отложить след феромона. Объяснение этому - то, что таким образом «привилегированный маршрут» запоминается в матрице следов феромона, и будущие муравьи будут использовать эту информацию, чтобы произвести новые решения в соседстве этого привилегированного маршрута. $\tau_{ij} = (1-a) \cdot \tau_{ij} + a / L_{best}$ (2) где L_{best} - самый короткий путь, полученный с начала вычисления (путь самого лучшего муравья).

След феромона также испаряется в течение построения решения. В этом случае, однако, он удален из посещенных ребер. Другими словами, каждый муравей, перемещаясь из города i в город j, применяет правило испарения феромона, которое заставляет количество следа феромона на ребре (i,j) уменьшаться. Правило: $\tau_{ij} = (1-a) \cdot \tau_{ij} + a \cdot \tau_0$ (3) где τ_0 - первоначальное значение следов. Считается, что хорошее значение для этого параметра $\tau_0 = (FirstSolution \cdot n)^{-1}$ (4) Объяснение использования формулы (3) – то, что муравьи съедают след феромона, в то время как они строят решения так, чтобы определенное множество в произведенных решениях было точно. (если бы след феромона не потреблялся муравьями, то они имели бы тенденцию производить очень подобные туры).

4. Локальный поиск для задач SOP

4.1 Эвристика обмена ребрами

В последние годы многие исследования были направлены на определение специальной TSP эвристики, см. (Johnson и McGeoch, 1997) для обзора. Они могут быть классифицированы как тур-конструирующая эвристика и тур-улучшающая эвристика (последнюю также называют эвристикой локальной оптимизации). Тур-конструирующая эвристика, см. Bentley (1992) для обзора, как правило, начинается выбором случайного города из набора городов, а затем постепенным построением возможного решения TSP путем добавления новых городов, выбранных согласно некоторому эвристическому правилу. Например, эвристика ближайшего соседства строит тур добавлением ближайшего города, с точки зрения расстояния, до последнего города, вставляемого в путь. С другой стороны, тур-улучшающая эвристика начинает с данного тура и пытается уменьшить его длину путем обмена ребрами, выбираемыми согласно некоторому эвристическому правилу, пока локальный оптимум находится (т.е., пока дальнейшее улучшение возможно с помощью эвристического правила). Было экспериментально показано (Reinelt, 1994), что, в общем, тур-улучшающая эвристика получает более качественные результаты, чем тур-конструирующая эвристика. Тем не менее, тур-конструирующая эвристика необходима, по крайней мере, для создания начального решениия для тур-улучшающей эвристики.

Начиная с начального решения, процедура обмена ребрами генерирует новое решение, заменяя k ребра другим набором k ребер. Эту операцию обычно называют k-обменом и многократно выполняют, пока никакой дополнительный улучшающий k-обмен не возможен.

В случае конечного решения оно считается k-оптимальным; проверка k-оптимальности требует в $O(n^k)$ раз. Для эффективности процедуры k-обмена необходимо, чтобы улучшающий критерий для новых решений мог быть вычислен в конечное время.

Было показано, что увеличение k производит решения повышенного качества, но вычислительные затраты полной проверки набора k-обмена для данного решения обычно ограничивают наше внимание k-обменом с k 3. Наиболее широко используемые процедуры обмена ребрами устанавливают k в 2 или 3 (2-opt и 3-opt процедуры обмена ребрами (Lin, 1965)), или в переменное значение (Lin и Kernighan, 1973), когда поиск переменной глубины обмена ребрами выполнен.

В этом разделе мы сначала строим некоторые соображения о методах обмена ребрами для TSP/ATSP задач. Потом мы концентрируем наше внимание на “пути, сохраняющем обмен ребрами” для ATSP, то есть на обмене ребрами, который не инвертируют порядок, в котором посещены пути. Затем мы обсуждаем «лексикографический путь, сохраняющий обмен ребрами», процедуру «пути, сохраняющего обмен ребрами», которая ищет только в пространстве допустимых обменов. Тогда мы добавляем к «лексикографическому пути, сохраняющему обмен ребрами» процедуру маркировки, функцию, которая должна проверить выполнимость в конечное время. Наконец, мы представляем различные возможные стратегии выбора узлов в течение поиска, а также различные критерии остановки поиска.

4.2 Эвристика пути, сохраняющего обмен ребрами

K-обмен удаляет k ребра из начального решения, создающего k несвязанные пути, которые повторно связаны с k новыми ребрами. В некоторых ситуациях эта операция требует поменять порядок, в котором посещены узлы в одном из путей («путь, инвертирующий обмен ребрами»), в то время как в других ситуациях, эта инверсия не требуется («путь, сохраняющий обмен ребрами»).

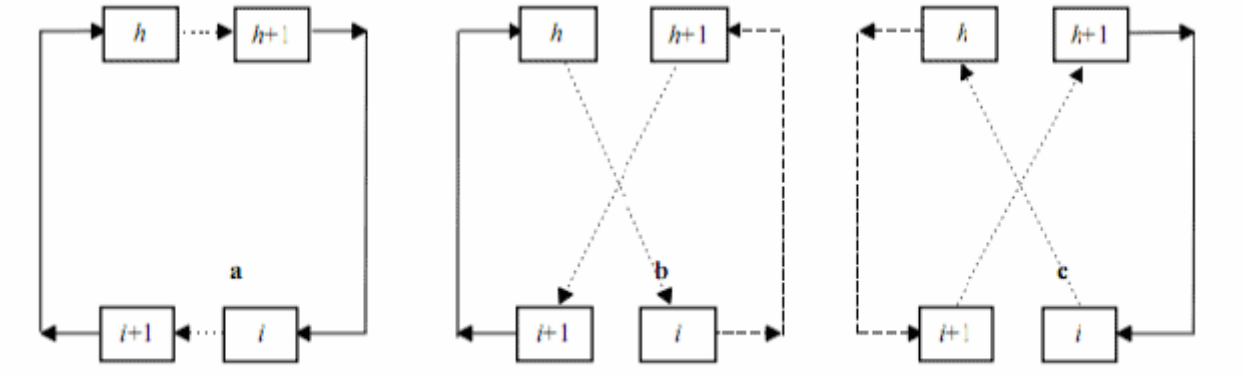

Рисунок 2. 2-обмен всегда инвертирует путь

Рассмотрим 2-обмен (рис. 2), где выбраны два ребра, которые будут удалены {h,h+1} и {i,i+1}. В этой ситуации есть только два способа выполнить обмен: в первом случае (рис. 2b) ребра {h,i} и {h+1,i+1} вставлены, и направление путешествия для пути (i,...,h+l) инвертировано; во втором случае (рис. 2c) ребра {i, h} и {i+1,h+1} вставлены, инвертируя направление путешествия для пути (h,...,i+1). В случае 3-обмена, однако, есть несколько возможностей построить новое решение, когда ребра {h,h+1}, {i,i+1}, и {j,j+1} отобраны для удаления (рис. 3). На рисунке 3 показаны: путь, сохраняющий 3-обмен (рис. 3b), и путь, инвертирующий 3-обмен (рис. 3c). Тогда понятно, что любая процедура 2-обмена определяет инверсию одного из включенных путей, в то время как для k=3 эта инверсия вызвана только отдельными выборами вставленных ребер.

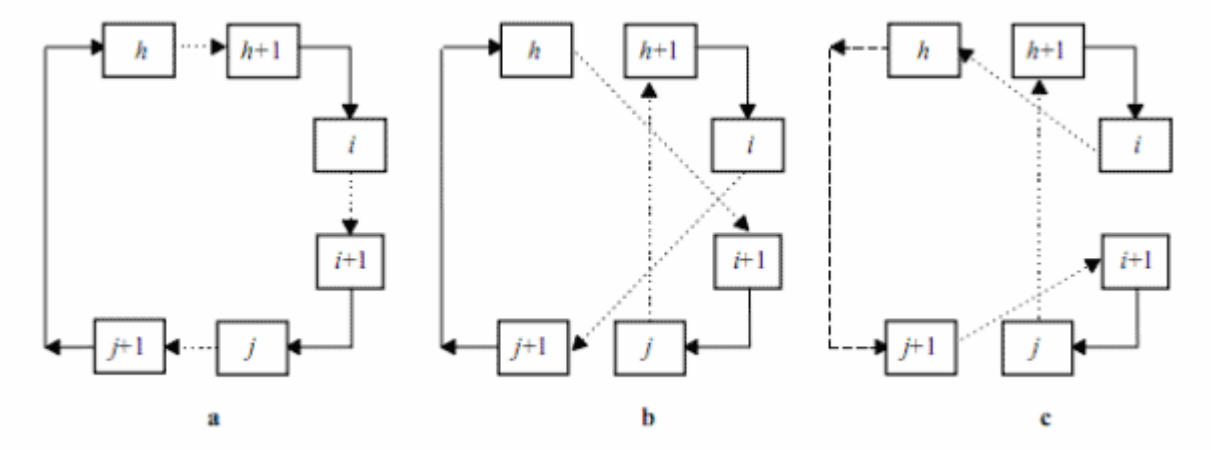

Рисунок 3. 3-обмен без (b) и с (c) инверсией пути

В случае задач TSP, где затраты = $\eta_{ij}=\eta_{ji}\forall(i,j)$, инвертируемый путь не изменяет свою длину. Поэтому, качество нового решения зависит только от длины вставленных и удаленных ребер. С другой стороны, для задач ATSP, где $\eta_{ij}\neq\eta_{ji}$, по крайней мере, для одного (i, j), инвертируя путь, можно изменить длину самого пути, и поэтому длина нового решения зависит не только от вставленных и удаленных ребер. Эта ситуация противоположна требованию, что улучшающий критерий будет проверяться в конечное время. Поэтому единственными подходящими процедурами обмена ребрами для задач SOP, которые являются ограниченной версией задач ATSP, являются эвристики “пути, сохраняющего обмен ребрами”. В дальнейшем, мы концентрируемся на “пути, сохраняющем k-обмен”, pp-k-exchange для краткости, с k=3, то есть, наименьшим k, которое позволяет “путь, сохраняющий обмен ребрами”.

4.3 Учет ограничений предшествования

Начиная с выполнимой SOP последовательности H, pp-3-exchange пытается уменьшить длину H заменой ребер {h,h+1}, {i,i+1} и {j,j+1} ребрами {h,i+1}, {i,j+l} и {j,h+1} (см. рис. 4a).

Результат pp-3-exchange - новая последовательность H (рис. 4b), где мы идем от узла 0 к узлу n, порядок, в котором мы посещаем путь path_left = (h+1,..., i) и path_right = (i+1,..., j), обратный.

В этой ситуации новая последовательность H является возможной, только если в начальном решении H между родительским узлом l path_left и родительским узлом r path_right нет никаких ограничений предшествования.

Учитывая два родительских пути path_left и path_right, чтобы проверить выполнимость pp-3-exchange требуются вычислительные затраты порядка $O(n^2)$ (ограничения предшествования должны быть проверены между каждой парой узлов в двух путях).

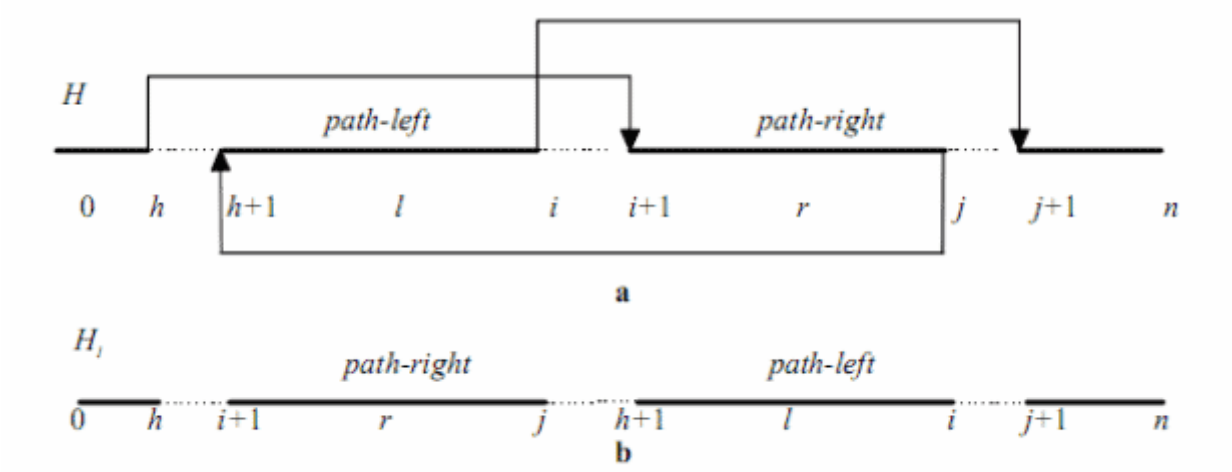

Рисунок 4. Путь, сохраняющий 3-обмен

Как ограничивать вычислительные затраты, необходимые для проверки выполнимости решения в случае ограничений предшествования, было изучено для задач dial-a-ride Сейвлсбергом (1990), который ввел отдельную стратегию исследования, названную лексикографической стратегией поиска, которая позволяет генерировать и исследовать только возможные обмены. Сейвлсберг предлагает комбинацию упомянутой лексикографической стратегии поиска с процедурой маркировки, где набор глобальных переменных модернизирован так, чтобы проверка ограничений предшествования могла быть выполнена в конечное время.

Лексикографическая стратегия поиска была введена для решения задач dial-a-ride, где позволяется только одно ограничение предшествования для каждого узла. Однако она не зависит от числа ограничений. Мы применили вариант лексикографической стратегии поиска Сейвлсберга, ограниченной случаем k=3, lpp-3-exchange, к задаче SOP с многократными ограничениями для каждого узла.

Процедура маркировки Сейвлсберга, однако, была разработана для учета уникальных ограничений предшествования при определенных условиях поиска и не может быть расширена для задач SOP. Перед объяснением нашей новой процедуры маркировки для SOP, мы представляем lpp-3-exchange.

4.4 Стратегия лексикографического поиска в случае ограничений предшествования

Процедура lpp-3-exchange идентифицирует два пути, path_left и path_right, которые единственным инвертированием повышают новое возможное решение. Эти два пути первоначально составлены из одного единственного узла и расширяются добавлением одного нового узла на каждом шаге. Эта особенность позволяет легко проверить выполнимость решения, потому что условия предшествования должны быть проверены только для нового добавленного узла.

Мы различаем прямой и обратный lpp-3-exchange. Мы говорим о прямом lpp-3-exchange (f-lpp-3-exchange), когда в начальной последовательности $\pi(h) < \pi(i) < \pi(j)$ ($\pi(x)$ указывает положение узла x в последовательности), и обратном lpp-3-exchange (b-lpp-3-exchange), когда $\pi(j) < \pi(i) < \pi(h)$.

lpp-3-exchange начинается, устанавливая узел h=0, и затем перемещает h сквозь последовательность, пока узел n-2 не достигнут. Каждый раз, когда узел h отобран, процедура выполняет прямой и обратный лексикографический поиск того же самого h.

f-lpp-3-exchange процедура начинается установкой значения h; потом устанавливает значение i, которое определяет самый правый узел левого пути (path_left), и выполняется цикл на значение j, которое определяет самый правый узел правого пути (path_right)(рис. 5b и 5d): path_right=(i+1,...j) итерационно расширяется новым ребром {j,j+1}. Когда все возможные узлы добавлены в правый путь, левый путь расширяется новым ребром {i,i+1}

(рис. 5c), и поиск снова начинается в правом пути. Левый и правый пути при инициализации состоят только из элементов i=h+1 и j=i+1 (рис. 5a).

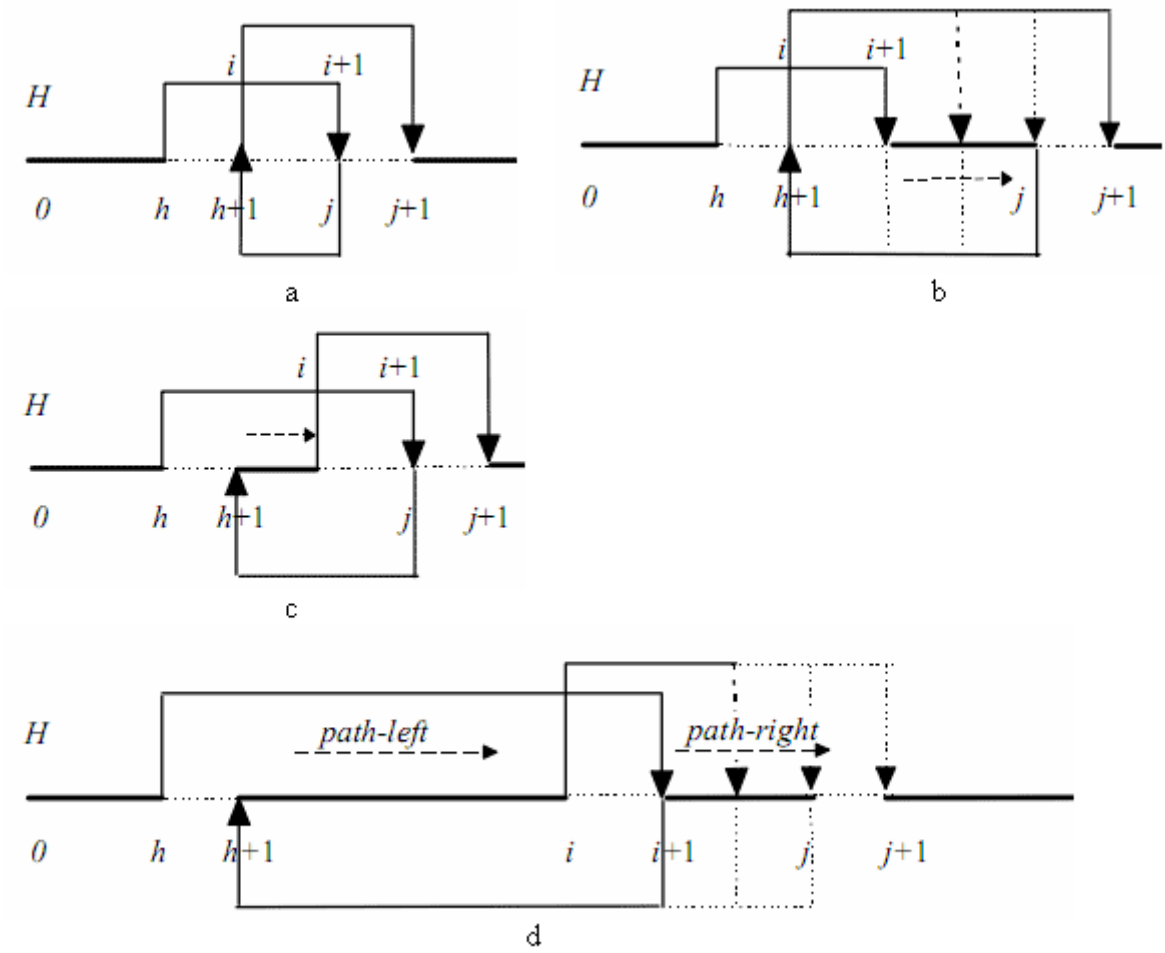

Рисунок 5. Прямой лексикографический “путь, сохраняющий 3-обмен”

В случае b-lpp-3-exchange (рис. 6) левый путь идентифицирован (j+1, ..., i) и правый (i+1, ..., h). После установки h, i=h−1 и j=i−1 (рис. 6a), и расширяем левый путь перемещением j до начала последовательности, присваивая j значения i−2, i−3, ..., 0 (рис. 6b). Затем правый путь итерационно расширяется в обратном порядке новым ребром {i,i+1}, и цикл в левом пути повторяется.

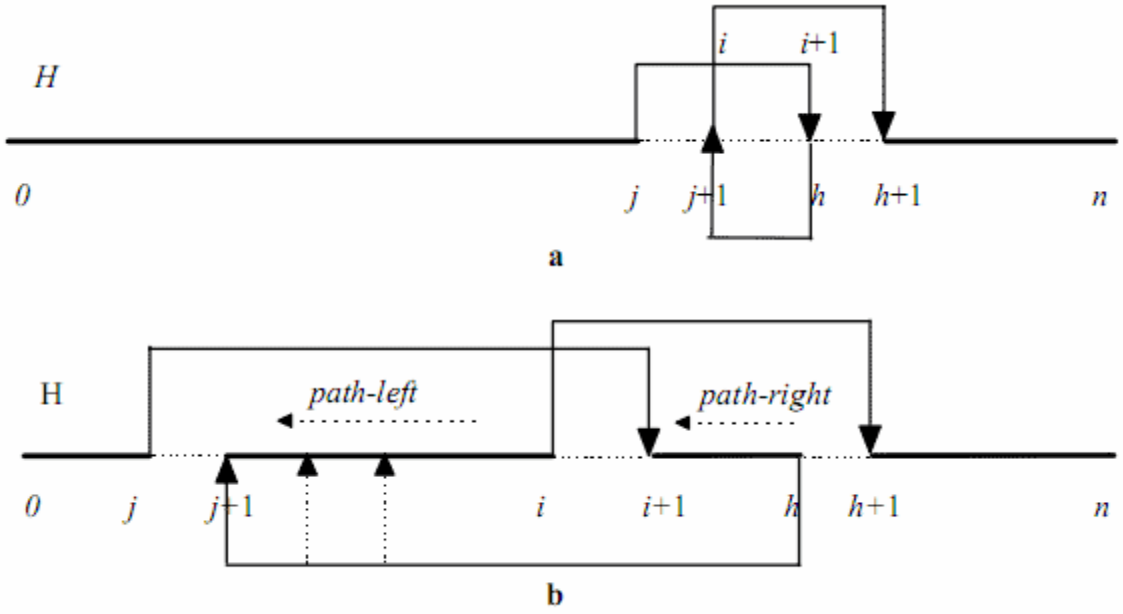

Рисунок 6. Обратный лексикографический “путь, сохраняющий 3-обмен”

Полный лексикографический поиск посещает все возможные узлы в последовательности (точно так же, как любая другая процедура 3-обмена), но метод для определения правого и левого путей полезен для легкого решения задачи проверки выполнимости: поиск ограничен только выполнимым обменом, так как он может быть остановлен, как только найден невыполнимый обмен. Рассмотрим, например, f-lpp-3-exchange: однажды path_left = (h+1, ..., i) был установлен, мы устанавливаем path_right в j=i+1. В этой ситуации возможно проверить выполнимость обмена, проверяя, если есть отношение предшествования между узлом j и узлами в path_left. Перед расширением правого пути новым ребром {j,j+1} мы проверяем, выполнимы ли все еще результирующие пути, проверкой отношения предшествования между новым узлом j+1 и узлами в левом пути. В случае, если тест не выполним, мы останавливаем поиск. Фактически, дальнейшее расширение j+1 в (j+2, j+3, ..., n) будет всегда генерировать невыполнимый обмен, потому что это нарушает, по крайней мере, ограничение предшествования между j+1 и path_left.

Заметьте, что расширение path_left ребром {i,i+1} не вызывает никаких нарушений ограничений предшествования, потому что порядок узлов внутри левого пути не изменен и поиск для выгодного f-lpp-3-exchange всегда начинается установкой правого пути равного элементу j=i+1.

Без рассмотрения никакой дополнительной процедуры маркировки, тест на выполнимость в этой ситуации имеет вычислительные затраты O(n): каждый раз, когда новое j выбрано, мы проверяем, если есть отношение предшествования между j и узлами в левом пути. В случае задач SOP тест должен проверить, будет ли $c_{jl} \neq -1 \ \forall l \in path\_left$ (вспомним, что для задач SOP $c_{ij}=-1$, если l должен предшествовать j, и в конечном решении H_1 порядок, в котором мы посещаем path_left и path_nght, инвертирован и поэтому l будет следовать за j, см. рис. 4b).

Подобные соображения должны быть сделаны в случае b-lpp-3-exchange, где тест выполнимости проверяет, если $c_{rj+1} \neq -1 \ \forall r \in path\_right$.

Предыдущая процедура полного лексикографического поиска требует проверки всех предшественников/преемников узла j. Эта процедура увеличивает вычислительную сложность проверки 3-оптимальности с $O(n^3)$ до $O(n^4)$.

Для сведения затрат к $O(n^3)$ мы вводим SOP маркирующую процедуру, способную учитывать многократные ограничения.

4.5 SOP маркирующая процедура

SOP маркирующая процедура используется, чтобы отметить узлы в последовательности маркером, который позволяет проверять выполнимость для каждого выбраного j в конечное время. Основная идея – привязать к каждому узлу маркер, который указывает, действительно ли возможно расширить path_right следующим узлом j+1.

Мы внедрили и испытали различные SOP маркирующие процедуры, которые устанавливают и обновляют узлы на различных этапах поиска. В дальнейшем мы представим сочетание самой эффективной SOP маркирующей процедуры со стратегией лексикографического поиска, с различными критериями выбора узла h и с различными критериями остановки поиска.

Наша SOP маркирующая процедура основана на множестве глобальных переменных, которые обновляются в течение процедуры лексикографического поиска. Как и в предыдущем подпункте, мы будем различать прямой и обратный поиск.

Во-первых, вводим глобальную переменную count_h, которая устанавливается в 0 в

начале поиска и увеличивается на 1 каждый раз, когда выбирается новый узел h. Во-вторых, привязываем глобальную переменную f-mark(v) к каждому узлу v∈H, в случае прямого лексикографического поиска (f-lpp-3-exchange), и глобальную переменную b-mark(v), в случае обратного лексикографического поиска (b-lpp-3-exchange). Эта глобальная переменная первоначально устанавливается в 0 для каждого v.

f-lpp-3-exchange начинается установкой h, i=h+1, и path_left=(i). В этом отношении, для всех узлов s∈successor[i] (successor[] – массив преемников) устанавливаем f-mark(s)=count_h. Повторяем эту операцию, каждый раз, когда path_left расширен новым узлом j. Процедура маркировки отмечает значением count_h все узлы в последовательности, которые должны следовать за одним из узлов, принадлежащих path_left. Когда path_right расширен, перемещаем j в (i+2,..., n), если f-mark (j) =count_h, останавливаем поиск, потому что маркер указывает, что j должен следовать за узлом в path_left. В этом отношении, если никакое другое условие завершения поиска не встречено, процедура повторно начинает снова расширять path_left. В этой ситуации все предыдущие определенные маркеры остаются действительными, и поиск продолжается, маркируя всех преемников нового i.

С другой стороны, перемещая h вперед в последовательность установкой count_h:=count_h+1, лишаются законной силы все предварительно установленные маркеры.

Тот же самый тип рассуждения действителен в случае b-lpp-3-exchange. Каждый раз, когда узел i отобран, идентифицируем новый path_right = (i+1,..., h), и для всех узлов s∈successor [i+1] устанавливаем b_mark(s) =count_h.

При расширении path_left многократным добавлением нового ребра {j,j+1}, расширение не происходит, если b_mark(j+1) =count_h.

4.5 Эвристика для выбора узла h и критерия остановки поиска

Представленная процедура последовательного поиска для задач SOP является общим описанием того, как лексикографический поиск работает в комбинации с SOP маркирующей процедурой.

Хотя SOP маркирующая процедура уменьшает сложность лексикографического поиска до $O(n^3)$, это все еще слишком дорого с практической точки зрения; фактически, все еще требуется исследование всех возможных обменов. Есть различные способы уменьшить эту работу: например, можно ввести некоторые эвристические критерии для уменьшения числа посещаемых узлов, или можно остановить поиск и выполнить обмен, как только встречено некоторое улучшающееся условие.

Эвристический выбор узла h. Чтобы уменьшить число исследуемых узлов, Сейвлсберг (1990) и Ван де Брагген и др. (1993) предлагают использовать отдельный тип k-обмена, называемый OR – обмен (Or, 1976), который ограничивает выбор i среди трех ближайших узлов h. Практически, i отобран среди (h+1, h+2, h+3) в случае прямого обмена, и среди (h-1, h-2, h-3) в случае обратного обмена.

Иначе, мы уменьшаем число посещаемых узлов, вводя две эвристики, которые влияют на то, как выбран узел h: одна основана на don’t look bit структуре данных, введенной Bentley (Bentley, 1992), в то время, как другая основана на новой структуре данных, названной don’t push stack, введенной авторами.

Don’t look bit – структура данных, в которой бит связан с каждым узлом последовательности. В начале поиска выключены все биты. Бит, связанный с узлом h, включен, когда поиск улучшающего движения, начинающегося с h, не удается. Бит, связанный с узлом h, выключен снова, когда улучшающий обмен с участием h выполнен.

Использование don’t look bit способствует исследованию узлов, которые были вовлечены в выгодный обмен. Процедура поиска всегда посещает все узлы в последовательности, начиная с первого узла 0 до последнего узла n, но только узлы с выключенным don’t look bit рассматриваются в качестве кандидатов на узел h. Используя этот подход, процедура поиска пытается оптимизировать последовательность, всегда начинающуюся с крайнего левого узла, еще не посещенного, или такого, какой в последний раз был вовлечен в выгодный обмен.

Don’t push stack – структура данных, основанная на стеке, который содержит набор узлов h для отбора, связанный с определенной push-операцией.

В начале поиска стек инициализируется всеми узлами (то есть, он содержит n+1 элемент). В течение поиска узел h извлекается из стека, и исследуются выполнимые 3-обмены, начиная с h. В случае выполнения выгодного обмена шесть узлов, вовлеченных в этот обмен (то есть, j+1, i+1,h+1, j, i, h), выталкиваются в стек (если они еще не принадлежат ему). Используя эту эвристику, когда-нибудь выгодный обмен выполнится, начиная с узла h, главным узлом в don’t push stack остается узел h. Кроме того, мы ограничиваем максимальный размер стека до n+1 элемента.

Использование don'tpush stack дает следующие преимущества. Во-первых, поиск сосредоточен на соседстве последнего обмена: это было экспериментально показано для приведения к лучшему показателю, чем полученный использованием don’t look bit (см. раздел 5.1). Во вторых, выбор узла h не ограничен последовательным проходом через последовательность H. Это важная особенность, учитывая тот факт, что SOP маркирующая процедура разработана для работы со случайным выбором h. Фактически, она не требует, как в случае процедуры маркировки Сейвлсберга (Savelsbergh, 1990), учитывать действительную информацию маркировки, проходя через последовательность от одного h к следующему: новая маркировка начата, как только новый h выбран; это учитывает отбор h в любой позиции последовательности, не вводя дополнительные вычислительные затраты.

Критерии остановки. Число посещаемых узлов может быть уменьшено, останавливая поиск, как только найден улучшающий обмен. В наших экспериментах мы проверили три различных условия остановки: (ExchangeFirst=h,i,j) останавливает поиск, как только найден первый выполнимый обмен в h, i или j циклах соответственно. Мы также проверили стандартную стратегию исследования, где самый выгодный обмен отобран среди всех возможных обменов, но этот метод не представлен здесь, потому что полученные результаты значительно хуже при таком же данном количестве времени вычисления.

Выводы

Оптимизация муравьиной колонии – новый подход к распределенной комбинаторной оптимизации. До сих пор алгоритмы колонии муравьев успешно применялись для решения симметричной и асимметричных задачам коммивояжера (Dorigo, Maniezzo и Colorni, 1991, 1996; Dorigo и Gambardella, 1997), квадратичной задачи о назначениях (Gambardella, Taillard и Dorigo, 1997), задачи раскраски графа (Costa и Hertz, 1997) и задач телекоммуникационной маршрутизации (Di Caro и Dorigo, 1997; 1998). В этой статье мы представили применение гибридного алгоритма, HAS-SOP, который сочетает в себе алгоритм муравьиной колонии с локальным поиском для решения задачи последовательного упорядочения. HAS-SOP был оценен экспериментально на множестве тестовых задач, доступных на TSPLIB, и он был сравнен с MPO/AI (Chen и Smith, 1996), одной из наиболее эффективных эвристик для задачи последовательного упорядочения. Результаты показывают, что в дополнение к опережающему MPO/AI, как с точки зрения качества результатов, так и требуемого процессорного времени для их нахождения, HAS-SOP также был в состоянии улучшить большинство верхних границ для вышеупомянутого набора тестовых задач.

Литература

1. Aarts E. H. and J. K. Lenstra, 1997, Introduction, in Local Search in Combinatorial Optimization, Aarts E. H. and J. K. Lenstra (Eds.). Chichester: John Wiley & Sons, 1–17.
2. Ascheuer N., 1996. Hamiltonian Path Problems in the On-line Optimization of Flexible Manufacturing Systems. ZIB Technical Report TR 96–3.
3. Ascheuer N., 1997, Personal communication.
4. Ascheuer N., Escudero L. F., Grotschel M. and Stoer M., 1993, A Cutting Plane Approach to the Sequential Ordering Problem (with Applications to Job Scheduling in Manufactoring), SIAM Journal on Optimization 3,25–42.
5. Balas E., Fischetti M., Pulleyblank W.R., 1995, The Precedence-constrained Asymmetric Traveling Salesman Polytope, Mathematical Programming 65, 241–265.
6. Battiti R. and Tecchiolli G., 1994, The reactive tabu search, ORSA Journal on Computing, 6, 126–140.
7. Bentley J.L., 1992, Fast algorithms for geometric traveling salesman problem, ORSA Journal on Computing, vol. 4, pp. 387–411.
8. Chen S. and Smith S., 1996, S.F. Commonality and Genetic Algorithms. Carnegie Mellon University, The Robotic Institute, Technical Report CMU-RI-TR-96-27.
9. Connolly D. T., 1990, An improved annealing scheme for the QAP, European Journal of Operational Research, 46, 93–100.
10. Costa D. and A. Hertz, 1997, Ants Can Colour Graphs, Journal of the Operational Research Society, 48, 295–305.
11. Di Caro G. and M. Dorigo, 1997. AntNet: A Mobile Agents Approach to Adaptive Routing, Tech. Rep. IRIDIA/97-12, Universite Libre de Bruxelles, Belgium.
12. Di Caro G. and M. Dorigo, 1998. Mobile Agents for Adaptive Routing, Proceedings of the 31st Hawaii International Conference on System Sciences (HICSS-31), Big Island of Hawaii, January 6-9, 1998, in press.
13. Dorigo M., 1992, "Ottimizzazione, Apprendimento Automatico, ed Algoritmi Basati su Metafora Naturale (Optimization, Learning and Natural Algorithms)," Doctoral dissertation, Dipartimento Elettronica e Informazione, Politecnico di Milano, Italy (in Italian).
14. Dorigo M., V. Maniezzo and A. Colorni, 1991, The Ant System: An Autocatalytic Optimizing Process. Tech. Rep. No. 91-016, Politecnico di Milano, Italy.
15. Dorigo M., Maniezzo V. and Colorni A., 1996, The Ant System: Optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics – Part B 26:(1), 29–41.
16. Dorigo M. and Gambardella L. M., 1997, Ant colony system: A cooperative Learning Approach to the Traveling Salesman Problem, IEEE Transactions on Evolutionary Computation 1,53–66.
17. Escudero L. F., 1988. An Inexact Algorithm for the Sequential Ordering Problem. European Journal of Operational Research 37, 232–253.
18. Escudero L.F., Guignard M., Malik K, 1994. A Lagrangian Relax-and-cut Approach for the Sequential Ordering Problem with precedence Relationships, Annals of Operations Research 50, 219–237.
19. Fleurent C. and Ferland J., 1994, Genetic hybrids for the quadratic assignment problem, DIMACS Series in Mathematical Theoretical Computer Science 16, 190–206.
20. Gambardella L. M., E. D. Taillard and M. Dorigo, 1997, Ant Colonies for the QAP. Tech. Rep. IDSIA/4-97, IDSIA, Lugano, Switzerland.
21. Kindervater G.A.P. and Savelsbergh M. W. P., 1997, Vehicle Routing: Handling Edge Exchanges, In Local Search in Combinatorial Optimization, Aarts E. H. and J. K. Lenstra (Eds.). Chichester: John Wiley & Sons, 311–336.
22. Johnson D.S. and McGeoch L.A., 1997, The Traveling Salesman Problem; a Case Study, In Local Search in Combinatorial Optimization, Aarts E. H. and J. K. Lenstra (Eds.). Chichester: John Wiley & Sons, 215–310.

23. Lin S., 1965, Computer solutions of the traveling salesman problem, Bell Systems Journal, vol. 44, pp. 2245–2269.
24. Lin S. and Kernighan B.W., 1973, An effective heuristic algorithm for the traveling salesman problem, Operations Research, vol. 21, pp. 498–516. Or I., 1976, Traveling Salesman-Type Combinatorial Problems and their Relation to the Logistics of Blood Banking, Ph.D. Thesis, Dept. of Industrial and Engineering and Management Science, Northwestern University, Evanston.
25. Psaraftis H.N., 1983, K-interchange Procedures for local Search in a precedence-Constrained Routing Problem, European Journal of Operational Research 13, 341–402.
26. Pulleyblank W. and M. Timlin, 1991, Precedence Constrained Routing and Helicopter Scheduling: Heuristic Design. IBM Technical Report RC17154 (#76032).
27. Reinelt G., 1994, The traveling salesman: computational solutions for TSP applications. Springer-Verlag.
28. Savelsbergh M. W. P., 1990, An efficient implementation of local search algorithms for constrained routing problems. European Journal of Operational Research 47, 75–85.
29. Solomon M.M., 1987, Algorithms for the Vehicle Routing and Scheduling Problems with Time Windows Constraints, Journal of Operational Research 35, 254–265.
30. Taillard E. D., 1991, Robust taboo search for the quadratic assignment problem, Parallel Computing, 17, 443–455.
31. Van der Bruggen L.J.J., Lenstra L.K., Schuur P.C., 1993, Variable-depth Search for the Single-Vehicle Pickup and Delivery Problem with Time Windows, Transportation Science 27, 298–311.