Ярослав Салий

# *Спортивное программирование*

**трудных (NP) задач**

*дискретной оптимизации*

01.04.2019 Семинар-130

# Где я? Кто все эти люди?

ПРИКЛАДНАЯ

ДИСКРЕТНАЯ

КОМБИНАТОРНАЯ

ОПТИМИЗАЦИЯ

МАТЕМАТИЧЕСКОЕ ПРОГРАММИРОВАНИЕ†

ЛИНЕЙНОЕ †

ЦЕЛОЧИСЛЕННОЕ †

ИССЛЕДОВАНИЕ ОПЕРАЦИЙ*

*мета*ЭВРИСТИКИ

ПРИБЛИЖЕННЫЕ СХЕМЫ

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ

БИБЛИОТЕКА ИСХОДНЫХ ДАННЫХ

ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ

*Operations Research (US)
Operational Research (UK)

† Mixed Integer-Linear Programming

# Где я? Кто все эти люди?

ПРИКЛАДНАЯ ДИСКРЕТНАЯ ОПТИМИЗАЦИЯ

КОМБИНАТОРНАЯ

МАТЕМАТИЧЕСКОЕ ПРОГРАММИРОВАНИЕ†

ЛИНЕЙНОЕ †

ЦЕЛОЧИСЛЕННОЕ †

ИССЛЕДОВАНИЕ ОПЕРАЦИЙ*

*мета*ЭВРИСТИКИ

ПРИБЛИЖЕННЫЕ СХЕМЫ

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ

БИБЛИОТЕКА ИСХОДНЫХ ДАННЫХ

ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ

*__Operations__ Research (US)
__Operational__ Research (UK)

† Mixed **Integer-Linear** Programming

# Где я? Кто все эти люди?

ПРИКЛАДНАЯ ДИСКРЕТНАЯ ОПТИМИЗАЦИЯ

КОМБИНАТОРНАЯ

МАТЕМАТИЧЕСКОЕ ПРОГРАММИРОВАНИЕ†

ЛИНЕЙНОЕ †

ЦЕЛОЧИСЛЕННОЕ †

ИССЛЕДОВАНИЕ ОПЕРАЦИЙ*

*мета*ЭВРИСТИКИ

ПРИБЛИЖЕННЫЕ СХЕМЫ

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ

БИБЛИОТЕКА ИСХОДНЫХ ДАННЫХ

ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ

***Operations** Research (US)
**Operational** Research (UK)

† Mixed **Integer-Linear** Programming

# Где я? Кто все эти люди?

ПРИКЛАДНАЯ

ДИСКРЕТНАЯ

КОМБИНАТОРНАЯ

ОПТИМИЗАЦИЯ

МАТЕМАТИЧЕСКОЕ ПРОГРАММИРОВАНИЕ†

ЛИНЕЙНОЕ †

ЦЕЛОЧИСЛЕННОЕ †

ИССЛЕДОВАНИЕ ОПЕРАЦИЙ*

*мета*ЭВРИСТИКИ

ПРИБЛИЖЕННЫЕ СХЕМЫ

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ

БИБЛИОТЕКА ИСХОДНЫХ ДАННЫХ

ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ

***Operations** Research (US)
**Operational** Research (UK)

† Mixed **Integer-Linear** Programming

# Где я? Кто все эти люди?

ПРИКЛАДНАЯ ДИСКРЕТНАЯ ОПТИМИЗАЦИЯ

КОМБИНАТОРНАЯ

МАТЕМАТИЧЕСКОЕ ПРОГРАММИРОВАНИЕ†

ЛИНЕЙНОЕ †

ЦЕЛОЧИСЛЕННОЕ †

ИССЛЕДОВАНИЕ ОПЕРАЦИЙ*

*мета*ЭВРИСТИКИ

ПРИБЛИЖЕННЫЕ СХЕМЫ

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ

БИБЛИОТЕКА ИСХОДНЫХ ДАННЫХ

ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ

***Operations** Research (US)
**Operational** Research (UK)

† Mixed **Integer-Linear** Programming

# Где я? Кто все эти люди?

**Operations Research**
**Mathematics of the Operations Research**
**Journal of the Operational Research Society**
**European Journal of Operational Research**
**Computers & Operations Research**
**Networks**
**Transportation Research part E**
**Omega**
**Computational Optimization and Applications**

Q1

Категории *Scimago Journal Rankings*:

- Management and Operations Research
- Computation Theory and Mathematics
- Discrete Mathematics and Combinatorics
- Theoretical Computer Science

The precedence-constrained asymmetric traveling salesman polytope☆ Mathematical Programming 68 (1995) 241–265

ДИСКРЕТНАЯ

ПРИКЛАДНАЯ

**A Branch & Cut Algorithm for the Asymmetric Traveling Salesman Problem with Precedence Constraints** Computational Optimization and Applications, 17, 61–84, 2000

КОМБИ

МАТЕМАТИЧЕСКОЕ ПРОГРАММИРОВАНИЕ†

**On Extended Formulations for the Precedence Constrained Asymmetric Traveling Salesman Problem** NETWORKS—2006—DOI 10.1002/net

**Multivalued Decision Diagrams for Sequencing Problems** OPERATIONS RESEARCH Vol. 61, No. 6, November–December 2013, pp. 1411–1428

ПРИБЛИЖЕННЫЕ СХЕМЫ

Load-dependent and precedence-based models for pickup and delivery problems Computers & Operations Research 63 (2015) 56–71

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ

БИБЛИОТЕКА ИСХОДНЫХ ДАННЫХ

An improved Ant Colony System for the Sequential Ordering Problem

ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ Computers and Operations Research 86 (2017) 1–17

Revisiting dynamic programming for precedence-constrained traveling salesman problem and its time-dependent generalization European Journal of Operational Research 272 (2019) 32–42

Operational Research (UK) Programming

# On $Spam_Eggs_Method for $Foo_Bar_Baz_Problem

1. Постановка задачи
   - Смежные постановки
   - Сложность (точного) решения
   - Кто еще решал эту задачу, чего добился?
2. Тестовые экземпляры общий аршин
3. Метод решения; препроцессинг
4. Вычислительный эксперимент
   - На чем считали? (CPU, OS, компилятор, спец.библиотеки)
   - Что померяли? (время счета, качество решения, расход памяти)
5. Авторы смогли…
   - *закрыть* X экземпляров … улучшить/достичь оценки сверху
6. Заключение
   - ДАЛЬШЕ! БОЛЬШЕ!! СИЛЬНЕЕ!!!

СТАТЬЯ НАПРИМЕР

# Неформальная постановка задачи

(введение)

So the precedence constraints are given by an acyclic and transitively closed digraph $P = (V, R)$. Using this notation we call a Hamiltonian path in $D_n$ *feasible* if $(j, i) \notin R$ holds for all $i < j$, where $i < j$ means that there is a directed path from node $i$ to node $j$ in the Hamiltonian path.

Now we can state the *sequential ordering problem* (SOP) formally. Given a complete digraph $D_n = (V, A_n)$ with costs $c_{ij}$ for all $(i, j) \in A_n$ and a transitively closed acyclic digraph $P = (V, R)$, find a feasible Hamiltonian path $\mathcal{H}$ in $D_n$ that has minimum cost.

objective is again to minimize the sum of setup times. As a benchmark problem, we consider the asymmetric traveling salesman problem with precedence constraints (ATSPP), also known as *sequential ordering problem*. The ATSPP is a variation of the asymmetric TSP where precedence constraints must be observed. Namely, given a weighted digraph $D = (V, A)$ and a set of pairs $P = V \times V$, the ATSPP is the problem of finding a minimum-weight Hamiltonian tour $T$ such that vertex $v$ precedes $u$ in $T$ if $(v, u) \in P$.

An instance of the SOP can be described using a graph $G = (V, E)$ where $V$ is a set of nodes containing the starting $u_s$ and the final $v_f$ nodes, and $E = \{(u, v) | u, v \in V, u \neq v\}$ is a set of weighted directed edges. Additionally, a precedence graph $H = (V, R)$ is given, where $R$ defines the precedence constraints, i.e. an edge $(u, v) \in R$ if node $u$ has to precede node $v$ in every *feasible* solution. By definition, the starting node $u_s$ precedes every other node, i.e. $(u_s, v) \in R \forall v \in V \backslash \{u_s\}$, and the final node $u_f$ has to be preceded by all other nodes, i.e. $(u, v_f) \in R \forall u \in V \backslash \{v_f\}$. The precedence graph, $R$, has to be acyclic for feasible solutions to exist.

*Operations Research (US)
Operational Research (UK)

† Mixed Integer-Linear Programming

# A cooperative parallel rollout algorithm for the sequential ordering problem

F. Guerriero *, M. Mancini

Parallel Computing 29 (2003) 663–677

Let $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ be a complete directed graph, with node set $\mathcal{N}$ and arc set $\mathcal{A}$. The nodes correspond to jobs $0, \ldots, i, \ldots, n$ ($|\mathcal{N}| = n + 1$) and it is assumed that the node 0 (i.e., starting node) and the node $n$ (i.e., final node) represent artificial jobs.

Each node $i \in \mathcal{N}$ is associated with the processing time $t_i$ of job $i$. The processing times of node 0 and node $n$ are set equal to 0.

To each arc $(i, j) \in \mathcal{A}$ is associated a cost-coefficient $c_{ij} \in \Re$, $c_{ij} \geqslant 0$, representing the waiting time between the end of job $i$ and the beginning of job $j$. The costs between the starting node 0 and the other nodes $i \in \mathcal{N} \setminus \{0, n\}$ are equal to the processing time of node $i$, $c_{0i} = t_i$, whereas the costs $c_{in}$, $\forall i \in \mathcal{N} \setminus \{0, n\}$ are equal to zero.

Precedence constraints are also given by an additional precedence digraph $\mathcal{P} = (\mathcal{N}, \mathcal{R})$ defined on the same node set $\mathcal{N}$. Each arc $(i, j) \in \mathcal{R}$ represents a precedence relationship, that is, in every feasible solution, job $i$ has to precede job $j$. It is worth noting that the precedence digraph $\mathcal{P} = (\mathcal{N}, \mathcal{R})$ must be acyclic. In addition, it can be assumed that if $i$ has to precede $j$ (i.e., $(i, j) \in \mathcal{R}$) and $j$ has to precede $k$ (i.e., $(j, k) \in \mathcal{R}$) then $i$ has to precede $k$ (i.e., $(i, k) \in \mathcal{R}$). In other words, the precedence digraph $\mathcal{P} = (\mathcal{N}, \mathcal{R})$ is transitively closed.

The $\mathcal{SOP}$ can be defined as the problem of finding a job sequence, starting at job 0 and ending at job $n$, that minimizes the total makespan subject to precedence constraints or, equivalently, the problem of finding a feasible Hamiltonian path, starting at node 0 and ending at node $n$, that satisfies the precedence constraints given by $\mathcal{P}$ with minimal total cost.

*Operatio
Operational Research (UK)

# TSPLIB—A Traveling Salesman Problem Library

GERHARD REINELT *Institut für Mathematik, Universität, Augsburg, Universitätsstrasse 8, D-8900 Federal Republic of Germany, CSNET: reinelt@augsopt.uni-augsburg.de*

ORSA Journal on Computing
Vol. 3, No. 4, Fall 1991

(Received: February 1991; accepted: May 1991)

https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

# A Branch & Cut Algorithm for the Asymmetric Traveling Salesman Problem with Precedence Constraints

Computational Optimization and Applications, 17, 61–84, 2000

NORBERT ASCHEUER* ascheuer@intranetz.de
*Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustr. 7, D-14195 Berlin-Dahlem, Germany*

MICHAEL JÜNGER mjuenger@informatik.uni-koeln.de
*Institut für Informatik der Universität zu Köln, Pohligstr. 1, D-50969 Köln, Germany*

GERHARD REINELT gerhard.reinelt@IWR.Uni-Heidelberg.De
*Institut für Angewandte Mathematik, Universität Heidelberg, Im Neuenheimer Feld 293, D-69120 Heidelberg, Germany*

*Operational Research (UK)

# A Branch & Cut Algorithm for the Asymmetric Traveling Salesman Problem with Precedence Constraints

Computational Optimization and Applications, 17, 61–84, 2000

**Real-life data.** The problem instances `ESC07`–`ESC98` are production data from IBM, that were supplied by Escudero. They correspond to the instances `P1`–`P9` as given in Ascheuer et al. [4]. The instances `rbg019a`–`rbg378a` correspond to data that was obtained from the routing of a stacker crane in an automatic storage system (for more details see Ascheuer [2]).

**Randomly generated problem instances.** All instances `prob.*` are randomly generated. `prob.7` (in TSPLIB `prob.100`) turned out to be one of the hardest instances we considered,

**Pure ATSP instances.** Since the ATSP is a special case of the SOP where the set of precedence relationships is empty, we obtain as a side product an algorithm to solve asymmetric TSPs to optimality. We run the algorithm with the (hard) ATSP instances contained

**ATSP instances with random precedences.** It turned out that the ATSPs that are the basis for the real-life problems `rbg*` are easily solvable. In order to check what influence additional precedence constraints have on hard ATSP instances we perform experiments on the ATSP instances contained in TSPLIB [24]. For that purpose we randomly generate precedence digraphs $P = (V, R)$ and add them to the ATSP instances.

```
NAME: ESC07.sop
TYPE: SOP
COMMENT: Received by Norbert Ascheuer / Laureano Escudero
DIMENSION: 9
EDGE_WEIGHT_TYPE: EXPLICIT
EDGE_WEIGHT_FORMAT: FULL_MATRIX
EDGE_WEIGHT_SECTION
9
    0    0    0    0    0    0    0    0 1000000
   -1    0  100  200   75    0  300  100    0
   -1  400    0  500  325  400  600    0    0
   -1  700  800    0  550  700  900  800    0
   -1   -1  250  225    0  275  525  250    0
   -1   -1  100  200   -1    0   -1   -1    0
   -1   -1 1100 1200 1075 1000    0 1100    0
   -1   -1    0  500  325  400  600    0    0
   -1   -1   -1   -1   -1   -1   -1   -1    0
EOF
```

*Operations Research (US)
Operational Research (UK)

Mixed Integer-Linear Programming

# New Benchmark Instances for Foo Bar Baz Problem

Awesome Optimization and Operations Research Vol. 14, No. 7, 2019, pp. 1423–1500

**Задачи на реальных данных**

**Случайно сгенерированные задачи**

**Близкородственные задачи**

**Близкородственные задачи, дополненные до Foo Bar Baz Problem**

исходные данные в формате AMPL доступны по ссылке:
http://natribu.org/de/FBBPLIB/instances

# New Benchmark Instances for Generalized TSP-PC

Awesome Optimization and Operations Research Vol. ??, No. ?, 2019, pp. ??–??

**Задачи на реальных данных.** Сверление, листовая резка, логистика, демонтаж энергоблоков АЭС

**Случайно сгенерированные задачи.** Города в мегаполисах на окружности, центры — случайно (равномерно) брошены на *евклидову* плоскость. Случайные условия предшествования

**Близкородственные задачи.** TSP-PC из TSPLIB, SOPLIB. Экземпляры GTSP Карапетяна
http://www.cs.nott.ac.uk/~pszdk/gtsp.html

**Близкородственные задачи, дополненные до GTSP-PC**
Кластеризация TSP-PC. Случайные условия предшествования для GTSP

**6. Outline of the implementations.** We have made three new implementations of cutting plane algorithms that compute lower bounds for the SOP. Two algorithms use model (2.5) and one uses model (2.4). Moreover, we compared this with the algorithm for the LP relaxation of model (2.3) described in [4].

Implementation A is based on model (2.4) and implementation B is based on model (2.5). Both were coded in FORTRAN, used Marsten's simplex-based LP solver XMP (see [15]), and were implemented and executed on a SIEMENS PC MX-2 (a 0.7 MIPS personal computer with UNIX operating system).

Implementation C is based on model (2.5). It was coded in PL/I version 1.5, used the algorithmic tools of the LP-solver MPSX (see [14]), and was implemented and executed on an IBM 4381 (a 7.7 MIPS computer with VM/CMS operating system).

TABLE 2

*Performance of our cutting separation implementations.*

| Case | $n$ | $\|R\|$ | Objective function value | | | | | Gap in objective function | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | H | E | A | B | C | E | A | B | C |
| *P1* | 7 | 7 | 2125 | 1950 | 2125 | 2125 | 2075 | 8.97 | 0.00 | 0.00 | 2.40 |
| *P1A* | 7 | 0 | 550 | 450 | 550 | 550 | 550 | 22.22 | 0.00 | 0.00 | 0.00 |
| *P2* | 11 | 5 | 2075 | 2021 | 2075 | 2075 | 2075 | 2.70 | 0.00 | 0.00 | 0.00 |
| *P2A* | 11 | 0 | 1866 | 1763 | 1866 | 1866 | 1843 | 5.80 | 0.00 | 0.00 | 1.25 |
| *P3* | 12 | 11 | 1675 | 1417 | 1598 | 1597 | 1535 | 18.20 | 4.82 | 4.88 | 9.12 |
| *P3A* | 12 | 0 | 1472 | 1386 | 1472 | 1472 | 1459 | 6.20 | 0.00 | 0.00 | 0.89 |
| *P4* | 14 | 14 | 2125 | 1525 | 2125 | 2125 | 2075 | 39.34 | 0.00 | 0.00 | 2.40 |
| *P5* | 25 | 11 | 1684 | 1518 | 1588 | 1577 | 1584 | 10.90 | 6.05 | 6.79 | 6.31 |
| *P5A* | 25 | 0 | 1145 | 1041 | 1134 | 1141 | 1118 | 10.00 | 0.97 | 0.35 | 2.42 |
| *P6* | 47 | 32 | 1288 | 1199 | 1219 | 1218 | 1219 | 7.40 | 5.66 | 5.75 | 5.66 |
| *P6A* | 47 | 0 | 915 | 856 | 872 | 872 | 871 | 6.09 | 4.93 | 4.93 | 5.05 |
| *P7* | 63 | 233 | 63 | 63 | 62 | 62 | 63 | 0.00 | 1.61 | 1.61 | 0.00 |
| *P7A* | 63 | 0 | 45 | 45 | 45 | 45 | 45 | 0.00 | 0.00 | 0.00 | 0.00 |
| *P8* | 78 | 283 | 18480 | 18205 | 18205 | 18205 | 18205 | 1.51 | 1.51 | 1.51 | 1.51 |
| *P8A* | 78 | 0 | 1845 | 1410 | 1305 | 1845 | 1712 | 30.85 | 41.37 | 0.00 | 7.76 |
| *P9* | 98 | 98 | 2125 | 1525 | 2125 | 2125 | 2075 | 39.34 | 0.00 | 0.00 | 2.40 |

Table 3 reports the CPU time required by the implementations. Implementations H, E, and C were run on an IBM 4381 and the time is given in seconds. Implementations A and B were run on a SIEMENS PC MX-2 and the time is given in minutes. All times reported include input-output operations. Note that the PC-implementation B solves the cases in less than $2\frac{1}{2}$ CPU hours. The mainframe version does this in a few seconds.

TABLE 3

CPU *time of our cutting separation implementations.*

| Case | $n$ | $\|R\|$ | H (sec.) | E (sec.) | A (min.) | B (min.) | C (sec.) |
|---|---|---|---|---|---|---|---|
| *P*1 | 7 | 7 | 0.08 | 0.10 | 0.14 | 0.10 | 0.05 |
| *P*1*A* | 7 | 0 | 0.13 | 0.24 | 0.18 | 0.10 | 0.05 |
| *P*2 | 11 | 5 | 0.26 | 0.55 | 0.17 | 0.20 | 0.04 |
| *P*2*A* | 11 | 0 | 0.22 | 0.55 | 1.05 | 0.15 | 0.09 |
| *P*3 | 12 | 11 | 0.16 | 0.36 | 9.59 | 1.27 | 0.89 |
| *P*3*A* | 12 | 0 | 0.25 | 0.56 | 1.06 | 0.18 | 0.21 |
| *P*4 | 14 | 14 | 0.31 | 1.10 | 0.17 | 0.10 | 0.71 |
| *P*5 | 25 | 11 | 1.17 | 1.19 | 23.34 | 0.43 | 0.55 |
| *P*5*A* | 25 | 0 | 0.35 | 1.28 | 13.25 | 0.46 | 0.45 |
| *P*6 | 47 | 32 | 3.05 | 4.78 | 87.39 | 4.51 | 1.14 |
| *P*6*A* | 47 | 0 | 1.98 | 3.85 | 100.10 | 1.20 | 1.08 |
| *P*7 | 63 | 233 | 4.35 | 3.38 | 340.43 | 27.27 | 5.63 |
| *P*7*A* | 63 | 0 | 0.06 | 3.47 | 109.37 | 1.42 | 4.08 |
| *P*8 | 78 | 283 | 27.62 | 12.93 | 443.13 | 153.01 | 12.05 |
| *P*8*A* | 78 | 0 | 8.93 | 6.94 | 250.52 | 9.36 | 18.41 |
| *P*9 | 98 | 98 | 28.47 | 25.01 | 0.23 | 0.20 | 6.20 |

TABLE 1. Results for the PCATS instances.

| | | | A | | B | | C | | D | | E | | F | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | V | P | Gap | CPU | Gap | CPU | Gap | CPU | Gap | CPU | Gap | CPU | Gap | CPU | Asch |
| ESC07 | 9 | 6 | 0.00 | 0 | 0.00 | 0 | 0.00 | 0 | 0.00 | 0 | 0.00 | 0 | 0.00 | 0 | 0.00 |
| ESC11 | 13 | 3 | 1.30 | 0 | 1.28 | 0 | 0.84 | 0 | 0.84 | 0 | 0.73 | 1 | 0.65 | 1 | 0.00 |
| ESC12 | 14 | 7 | 0.36 | 1 | 0.00 | 0 | 0.00 | 0 | 0.00 | 0 | 0.00 | 0 | 0.00 | 0 | 0.00 |
| ESC25 | 27 | 9 | 6.51 | 26 | 5.55 | 24 | 2.56 | 28 | 2.40 | 23 | 1.76 | 120 | 1.92 | 24 | 2.32 |
| ESC47 | 49 | 10 | 4.10 | 650 | 4.10 | 646 | 4.08 | 1011 | 4.08 | 719 | 3.46 | 3642 | 3.42 | 752 | 3.18 |
| p43.1 | 44 | 9 | 0.04 | 11512 | 0.01 | 1741 | 0.01 | 2209 | 0.01 | 779 | 0.01 | 1909 | 0.01 | 7374 | 0.27 |
| p43.2* | 44 | 20 | 0.62 | 53761 | 0.39 | 8140 | 0.60 | 22258 | 0.39 | 7016 | 0.35 | 16160 | 0.29 | 13567 | 1.30 |
| p43.3* | 44 | 37 | 1.43 | 22620 | 0.77 | 5096 | 1.43 | 3461 | 0.77 | 3786 | 0.72 | 8943 | 0.67 | 4579 | 2.34 |
| p43.4* | 44 | 58 | 16.51 | 743 | 16.42 | 578 | 16.46 | 448 | 16.42 | 344 | 16.41 | 1742 | 0.16 | 440 | 32.67 |
| ry48p.1 | 49 | 11 | 7.98 | 1500 | 3.40 | 1755 | 6.64 | 1179 | 3.03 | 1586 | 2.95 | 3835 | 3.00 | 2670 | 5.29 |
| ry48p.2* | 49 | 23 | 11.84 | 1840 | 7.03 | 1761 | 10.52 | 544 | 5.85 | 1643 | 5.76 | 3833 | 5.72 | 3478 | 9.29 |
| ry48p.3* | 49 | 42 | 14.84 | 1095 | 13.19 | 1029 | 14.02 | 898 | 12.80 | 1061 | 12.41 | 3719 | 10.94 | 2013 | 13.30 |
| ry48p.4* | 49 | 58 | 18.84 | 899 | 18.50 | 765 | 18.84 | 709 | 18.50 | 611 | 17.59 | 2351 | 12.87 | 720 | 13.61 |
| ft53.1 | 54 | 12 | 5.87 | 3281 | 3.46 | 3888 | 3.66 | 2373 | 2.76 | 2214 | 2.66 | 6991 | 2.29 | 9699 | 4.58 |
| ft53.2* | 54 | 25 | 10.20 | 3284 | 7.02 | 3764 | 7.56 | 1876 | 6.84 | 1851 | 6.66 | 5878 | 5.89 | 10780 | 7.88 |
| ft53.3* | 54 | 48 | 16.82 | 1953 | 14.58 | 1975 | 16.49 | 1180 | 14.48 | 1221 | 13.50 | 4253 | 9.94 | 3293 | 11.18 |
| ft53.4* | 54 | 63 | 7.76 | 1473 | 7.52 | 1221 | 7.54 | 936 | 7.31 | 994 | 5.58 | 4181 | 3.26 | 836 | 3.84 |

ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ

*Operations Research (US)
Operational Research (UK)

† Mixed Integer-Linear Programming

### 10.1. Experimental Setup

The experiments were performed on a computer equipped with an Intel Xeon E5345 at 2.33 GHz with 8 Gb RAM. The MDD code was implemented in C++ using the CPO callable library from the IBM ILOG CPLEX Academic Studio V.12.4.01. We have set the following additional CPO parameters for all experiments: `Workers` = 1, to use a single computer core; `DefaultInferenceLevel` = `Extended`, to use the maximum possible propagation available in CPO; and `SearchType` = `DepthFirst`.

**Table 1.** Results on ATSPP instances.

| | | | CPO | | CPO + MDD, width 2,048 | |
|---|---|---|---|---|---|---|
| Instance | Vertices | Bounds | Best | Time (s) | Best | Time (s) |
| br17.10 | 17 | 55 | 55 | 0.01 | 55 | 4.98 |
| br17.12 | 17 | 55 | 55 | 0.01 | 55 | 4.56 |
| ESC07 | 7 | 2,125 | 2,125 | 0.01 | 2,125 | 0.07 |
| ESC25 | 25 | 1,681 | 1,681 | TL | 1,681 | 48.42 |
| p43.1 | 43 | 28,140 | 28,205 | TL | 28,140 | 287.57 |
| p43.2 | 43 | [28,175, 28,480] | 28,545 | TL | **28,480** | **279.18** |
| p43.3 | 43 | [28,366, 28,835] | 28,930 | TL | **28,835** | **177.29** |
| p43.4 | 43 | 83,005 | 83,615 | TL | 83,005 | 88.45 |
| ry48p.1 | 48 | [15,220, 15,805] | 18,209 | TL | 16,561 | TL |
| ry48p.2 | 48 | [15,524, 16,666] | 18,649 | TL | 17,680 | TL |
| ry48p.3 | 48 | [18,156, 19,894] | 23,268 | TL | 22,311 | TL |
| ry48p.4 | 48 | [29,967, 31,446] | 34,502 | TL | **31,446** | **96.91** |
| ft53.1 | 53 | [7,438, 7,531] | 9,716 | TL | 9,216 | TL |
| ft53.2 | 53 | [7,630, 8,026] | 11,669 | TL | 11,484 | TL |
| ft53.3 | 53 | [9,473, 10,262] | 12,343 | TL | 11,937 | TL |
| ft53.4 | 53 | 14,425 | 16,018 | TL | 14,425 | 120.79 |

*Note.* Values in bold represent instances solved for the first time.

## 7. Experiments

This section reports and discusses experimental results for instances of the $m$-PDTSP and the SOP. Each test run was performed on a single core of an Intel Xeon E5540 or E5649 machine both with 2.53 GHz. Preliminary tests showed that both machines have nearly the same performance with respect to our type of experiments. The memory limit per test run was set to 8 GB. The CPU times for the preprocessing from Section 2 is included in all given running times.

Table 6 shows branch-and-cut results for instances for the SOP from the TSPLIB. The best known (BK) lower and upper bounds (LB,UB) and fastest solution times are obtained from different articles [3–5,11,16,8]. Note that the BK results are obtained on different hardware so they are not directly comparable to our CPU times. Dashes "–" in the tables mean a reached time or memory limit. We compare four different branch-and-cut configurations BC1–4: the heuristic separation and inequalities (24) are only active in BC1–2, we set $\Delta_G = 0.5$ for BC1/3 and $\Delta_G = 0.9$ for BC2/4. Lower and upper bounds BCx are the best over all four branch-and-cut algorithms.

**Table 6**
Comparison of branch-and-cut algorithms for SOP instances from the TSPLIB. Bold instance names mark instances solved for the first time. Bold bounds and CPU times denote the best results.

| Instance | $\|V\|$ | $\|A\|$ | $\|\bar{R}\|$ | LB | | UB | | Time in seconds | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | BK | BCx | BK | BCx | BK | BC1 | BC2 | BC3 | BC4 |
| ESC07 | 9 | 40 | 6 | 2125 | 2125 | 2125 | 2125 | 0 | 0 | 0 | 0 | 0 |
| ESC12 | 14 | 132 | 7 | 1675 | 1675 | 1675 | 1675 | 0 | 0 | 0 | 0 | 0 |
| ESC25 | 27 | 622 | 9 | 1681 | 1681 | 1681 | 1681 | 1 | **0** | **0** | **0** | **0** |
| ESC47 | 49 | 2187 | 10 | 1288 | 1288 | 1288 | 1288 | 28 | 7 | 8 | 4 | **2** |
| ESC63 | 65 | 3613 | 95 | 62 | 62 | 62 | 62 | **0** | 9 | 2 | 1 | 1 |
| ESC78 | 80 | 5550 | 77 | 18230 | 18230 | 18230 | 18230 | **1** | 770 | 46 | 8664 | 7569 |
| br17.10 | 18 | 237 | 10 | 55 | 55 | 55 | 55 | **0** | 1 | **0** | 1 | **0** |
| br17.12 | 18 | 223 | 12 | 55 | 55 | 55 | 55 | **0** | 1 | **0** | 1 | **0** |
| ft53.1 | 54 | 2722 | 12 | 7531 | 7531 | 7531 | 7531 | 6768 | 183 | 218 | **91** | 145 |
| **ft53.2** | 54 | 2680 | 25 | 7630 | **8026** | 8026 | 8026 | – | **29842** | – | – | – |
| **ft53.3** | 54 | 2306 | 48 | 9473 | **10262** | 10262 | 10262 | – | 15693 | **8629** | – | – |
| ft53.4 | 54 | 1218 | 63 | 14425 | 14425 | 14425 | 14425 | 121 | 11 | **2** | 5 | 5 |
| ft70.1 | 71 | 4783 | 17 | 39313 | 39313 | 39313 | 39313 | 363 | 27 | 48 | **17** | 18 |
| ft70.2 | 71 | 4714 | 35 | 39843 | **40101** | **40419** | 40728 | – | – | – | – | – |
| **ft70.3** | 71 | 4384 | 68 | 41413 | **42535** | 42535 | 42535 | – | 61197 | **28691** | – | – |
| **ft70.4** | 71 | 2154 | 86 | 53072 | **53530** | 53530 | 53530 | – | 769 | 315 | 308 | **249** |
| kro124p.1 | 101 | 9814 | 25 | 37861 | **38762** | 39420 | 39420 | – | – | – | – | – |
| kro124p.2 | 101 | 9738 | 49 | 38809 | **39841** | 41336 | 41336 | – | – | – | – | – |
| kro124p.3 | 101 | 9339 | 97 | 41578 | **43904** | **49499** | 49570 | – | – | – | – | – |
| kro124p.4 | 101 | 5260 | 131 | 65445 | **73021** | 76103 | 76103 | – | – | – | – | – |

## 5. Computational experiments

A series of computational experiments was conducted in order to evaluate the performance of the proposed algorithms. In the first part of the experiments we focused on the efficiency of the ACS-SA and EACS-SA used alone, i.e. without the problem-specific LS. In the second part the focus was placed on the efficiency of the algorithms coupled with the SOP-3-exchange and SOP-3-exchange-SA LS heuristics.

The ACS and EACS require that a number of parameters be set. Based on preliminary computations and suggestions from the literature we used the following settings in our experiments: number of ants, $m = 10$; $\beta = 0.5$; $\psi = 0.01$ and $\rho = 0.1$, and local and global pheromone evaporation ratio, respectively; $q_0 = \frac{n-20}{n}$, where $n$ is the size of the problem. The computations were repeated 30 times for each configuration of the parameter values and the problem instance. The computations were conducted on a machine equipped with a Xeon E5-2680v3 12 core CPU clocked at 2.5GHz, although a single core was used per run. All algorithms were implemented in C++ and compiled with the GNU compiler with the -Ofast switch.[1]

[1] The source code is available at https://github.com/RSkinderowicz/AntColonySystemSA.

**Table 2**
Summary of results obtained by the ACS, ACS-SA, EACS and EACS-SA algorithms on a set of 20 SOP instances from the TSPLIB repository. The left-most part of the table contains the mean solution lengths along with the standard deviations shown in the braces. The last 6 columns contain a summary of the statistical comparison between the respective pairs of algorithms according to the two-sided, non-parametric Bonferroni–Dunn test with a family-wise Type I error correction ($\alpha_{FW} = 0.05$). The capital letter indicates the algorithm which obtained significantly better results than the others. Hyphens denote that there were no significant differences between the results of the respective algorithms.

| Problem | ACS (A) | ACS-SA (B) | EACS (C) | EACS-SA (D) | A vs B | A vs C | A vs D | B vs C | B vs D | C vs D |
|---|---|---|---|---|---|---|---|---|---|---|
| ft53.1 | 7857 (161.8) | 7673 (18.5) | 7818 (162.9) | 7702 (46.1) | B | – | D | B | – | D |
| ft53.2 | 8713 (159.5) | 8522 (107.0) | 8647 (256.8) | 8348 (156.6) | B | – | D | – | D | D |
| ft53.3 | 11,506 (578.9) | 11,417 (506.4) | 11,271 (605.5) | 11,418 (544.0) | – | – | – | – | – | – |
| ft53.4 | 14,744 (201.8) | 14,704 (81.7) | 14,779 (128.2) | 14,639 (101.1) | – | – | D | – | D | D |
| ft70.1 | 40,437 (458.0) | 40,054 (223.6) | 40,692 (588.6) | 40,150 (345.5) | B | – | – | B | – | D |
| ft70.2 | 42,263 (454.5) | 41,629 (409.0) | 42,396 (664.4) | 41,710 (355.0) | B | – | D | B | – | D |
| ft70.3 | 44,674 (667.0) | 44,388 (333.3) | 44,589 (570.0) | 43,946 (436.6) | – | – | D | – | D | D |
| ft70.4 | 56,098 (325.5) | 56,146 (127.0) | 55,593 (564.1) | 55,305 (362.9) | – | C | D | C | D | – |
| kro124p.1 | 42,166 (757.8) | 41,313 (572.8) | 42,324 (988.9) | 41,768 (731.4) | B | – | – | B | – | – |
| kro124p.2 | 44,548 (1113.2) | 43,049 (764.4) | 44,270 (1318.3) | 43,529 (809.9) | B | – | D | B | – | – |
| kro124p.3 | 53,915 (1832.6) | 51,411 (321.2) | 53,313 (1678.5) | 51,351 (963.8) | B | – | D | B | – | D |
| kro124p.4 | 80,373 (1120.1) | 79,708 (922.3) | 81,204 (1064.1) | 78,973 (746.0) | – | – | D | B | – | D |
| prob.100 | 1485 (87.8) | 1489 (75.2) | 1505 (76.6) | 1438 (66.8) | – | – | – | – | D | D |
| rbg109a | 1111 (9.9) | 1107 (11.4) | 1093 (9.7) | 1067 (7.9) | – | C | D | C | D | D |
| rbg150a | 1872 (11.8) | 1855 (9.7) | 1817 (11.6) | 1788 (10.5) | – | C | D | C | D | D |
| rbg253a | 3156 (15.1) | 3123 (13.6) | 3101 (19.8) | 3041 (9.0) | B | C | D | – | D | D |
| rbg341a | 3103 (60.8) | 2989 (40.8) | 2990 (37.2) | 2821 (31.5) | B | C | D | – | D | D |
| rbg323a | 3590 (29.9) | 3517 (21.0) | 3540 (27.8) | 3393 (31.3) | B | C | D | – | D | D |
| rbg358a | 3284 (81.1) | 3128 (34.9) | 3087 (52.0) | 2883 (36.3) | B | C | D | – | D | D |
| rbg378a | 3483 (54.0) | 3347 (38.9) | 3442 (59.8) | 3184 (37.5) | B | – | D | B | D | D |

# Experimental Setup

Вычислительный эксперимент

1. Характеристики «установки»
   - хард — CPU, RAM, interconnect
   - софт — OS, компилятор/пакет, сторонние библиотеки
2. Пред. обработка исходных данных
3. Выходные данные
   - время счета — CPU time / wall-clock time; как меряли (`time.h::time()` / `std::chrono::…?`)
   - число запусков
   - результат — ВГР, НГР, «разбег» (GAP)



*Operations Research (US)
Operational Research (UK)

† Mixed Integer-Linear Programming

# А дальше?

ДИСКРЕТНАЯ ПРИКЛАДНАЯ ОПТИМИЗАЦИЯ КОМБИНАТОРНАЯ МАТЕМАТИЧЕСКОЕ ПРОГРАММИРОВАНИЕ† ЦЕЛОЧИСЛЕННОЕ† НЕЛОЧИСЛЕННОЕ† ИССЛЕДОВАНИЕ ОПЕРАЦИЙ* ЭВРИСТИКИ ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ БИБЛИОТЕКА ИСХОДНЫХ ДАННЫХ ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ

## TSPLIB и его друзья

https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

http://www.cs.nott.ac.uk/~pszdk/gtsp.html

http://www.idsia.ch/~roberto/SOPLIB06.zip

## DIMACS Implementation Challenges

http://archive.dimacs.rutgers.edu/Challenges/

*Operations Research (US)
Operational Research (UK)

† Mixed Integer-Linear Programming

# Спасибо за внимание!

ДИСКРЕТНАЯ

ПРИКЛАДНАЯ

ОПТИМИЗАЦИЯ

КОМБИНАТОРНАЯ

МАТЕМАТИЧЕСКОЕ ПРОГРАММИРОВАНИЕ†

ЛИНЕЙНОЕ †

ЦЕЛОЧИСЛЕННОЕ †

ИССЛЕДОВАНИЕ ОПЕРАЦИЙ*

*мета*ЭВРИСТИКИ

ПРИБЛИЖЕННЫЕ СХЕМЫ

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ

БИБЛИОТЕКА ИСХОДНЫХ ДАННЫХ

ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ

*__Operations__ Research (US)
__Operational__ Research (UK)

† Mixed __Integer-Linear__ Programming

ДИСКРЕТНАЯ

ПРИКЛАДНАЯ

ОПТИМИЗАЦИЯ

КОМБИНАТОРНАЯ

МАТЕМАТИЧЕСКОЕ ПРОГРАММИРОВАНИЕ†

ЛИНЕЙНОЕ †

ЦЕЛОЧИСЛЕННОЕ †

ИССЛЕДОВАНИЕ ОПЕРАЦИЙ*

*мета*ЭВРИСТИКИ

ПРИБЛИЖЕННЫЕ СХЕМЫ

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ

БИБЛИОТЕКА ИСХОДНЫХ ДАННЫХ

ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ

*Operations Research (US)
Operational Research (UK)

† Mixed Integer-Linear Programming

ДИСКРЕТНАЯ

ПРИКЛАДНАЯ

ОПТИМИЗАЦИЯ

КОМБИНАТОРНАЯ

МАТЕМАТИЧЕСКОЕ ПРОГРАММИРОВАНИЕ†

ЛИНЕЙНОЕ †

ЦЕЛОЧИСЛЕННОЕ †

ИССЛЕДОВАНИЕ ОПЕРАЦИЙ*

*мета*ЭВРИСТИКИ

ПРИБЛИЖЕННЫЕ СХЕМЫ

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ

БИБЛИОТЕКА ИСХОДНЫХ ДАННЫХ

ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ

*Operations Research (US)
Operational Research (UK)

† Mixed Integer-Linear Programming