A comparison of two exact algorithms for the sequential ordering problem

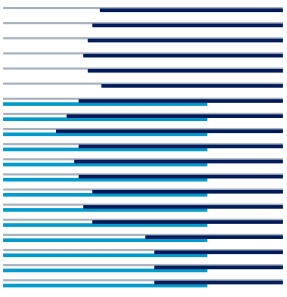
V. Papapanagiotou[a], J. Jamal[a], R. Montemanni[a], G. Shobaki[b], and L.M. Gambardella[a]

[a] *Dalle Molle Institute for Artificial Intelligence, USI/SUPSI Manno, Switzerland*

{vassilis, jafar, roberto, luca}@idsia.ch

[b] *California State University, Sacramento Sacramento, California, USA*

ghassan.shobaki@csus.edu



Technical Report No. IDSIA-12-15
December 2015

IDSIA / USI-SUPSI
Istituto Dalle Molle di studi sull'intelligenza artificiale
Galleria 2, 6928 Manno, Switzerland

A comparison of two exact algorithms for the sequential ordering problem

V. Papapanagiotou[a], J. Jamal[a], R. Montemanni[a], G. Shobaki[b], and L.M. Gambardella[a]

[a] *Dalle Molle Institute for Artificial Intelligence, USI/SUPSI Manno, Switzerland*
{*vassilis, jafar, roberto, luca*}*@idsia.ch*
[b] *California State University, Sacramento Sacramento, California, USA*
ghassan.shobaki@csus.edu

December 2015

Abstract

The sequential ordering problem is an NP-hard combinatorial optimization problem that arises in many realworld applications in logistics and scheduling. In this paper, we experimentally compare two exact algorithms that have been recently proposed for solving this problem. The comparison is done on a large set of benchmarks used in published work. In addition to comparing the performance of two different algorithms, this paper reports the closing of nine instances for the first time as well as seventeen improved upper and lower bounds.

INTRODUCTION

The Sequential Ordering Problem (SOP) is a combinatorial optimization problem that may be described as follows. Given a weighted directed graph, a set of precedence constraints between vertex pairs and a starting vertex, the objective is to find a minimum-cost Hamiltonian path. A path is feasible if it fulfills the precedence constraints. Without precedence constraints, the SOP is equivalent to an Asymmetric Travelling Salesman Problem (ATSP), and when the latter has symmetric costs, it is referred to as the Travelling Salesman Problem (TSP). Therefore, the SOP is a generalization of the TSP, and since the TSP is a well-known NP-hard problem, the SOP must be NP-hard. The problem has been initially formulated by Escudero [1] as the underlying model for a production planning system. Escudero also presents an inexact algorithm that exploits the properties of the ATSP poly-tope. A similar approach is taken by Ascheuer et al. [2], who show how to generate valid cuts using a polynomial-time separation algorithm. Another cuttingplane approach based on a Lagrangian relaxation was presented by Escurero et al. [3]. Hernadvolgy [4, 5] generates good lower bounds using a technique that reduces the instance size. Balas et al. [6] presents a deeper analysis of the impact of precedence constraints on the ATSP convex hull. Meta-heuristic methods have also been proposed for solving the SOP. Chen and Smith [7] and Moon et al. [8] propose genetic algorithms. Pulleyblank and Timlin [9] use a Voronoi quantized crossover that adopts a complete graph representation. Gambardella and Dorigo [10] describe the HAS-SOP algorithm, which couples an ant-colony system with a 3-opt local search. The effectiveness of this algorithm appears to be dependent on the density of the precedence constraints. Montemanni et al. [11, 12] show how to exploit this property by artificially manipulating the density of precedence constraints. Experiments comparing the previous two algorithms on a set of real problems arising in quay crane assignment have been reported by Montemanni et al. [13,

14]. Anghinolfi et al. [15, 16] propose a Particle Swarm Optimization (PSO) approach to solving the SOP. A metaheuristic approach combining a Mixed Integer Linear Programming (MILP) solver and an Ant System has been described by Mojana et al. [17]. In this paper, we present an experimental comparison between two exact algorithms that have been recently proposed for solving the SOP. The experimental evaluation is performed using a large set of benchmarks from published work. The first algorithm, which was proposed by Montemanni et al. [18, 19, 20], is based on MILP and problem decomposition. This algorithm will be referred to as DEC. The second algorithm, which was proposed by Shobaki and Jamal [21], is based on a branch-and-bound framework with a set of lower bound and pruning techniques. This algorithm will be referred to as B&B.

PROBLEM DESCRIPTION

The SOP can be modeled by a graph as follows. A complete weighted directed graph D = (V, A) is given, where V is the set of vertices and $A = \{(i, j)|i, j \in V\}$ is the set of arcs. A cost $c_{ij} \in {N_0}^+$ is associated with each arc (i, j) $\in$ A. Two given vertices of V are by definition the beginning and the end of the solution path. Furthermore, a precedence digraph P = (V, R) is given. P is defined on the same vertex set V as D. An arc (i, j) $\in$ R represents a precedence constraint, that is, vertex i has to precede vertex j in any feasible path. The precedence digraph P must be acyclic. A feasible solution is a Hamiltonian path (a path that includes each vertex exactly once) that satisfies the precedence constraints. The objective is to find a feasible solution with minimal total cost.

THE DECOMPOSITION-BASED APPROACH (DEC)

It is well known that for the SOP, the effort needed to close an instance does not only depend on the number of nodes but it also depends on the structure of the precedence graph. When the precedence graph is either very dense or very sparse, it is generally easier to solve a problem than when the graph has medium density. Moreover, some sub-structures in the precedence graph could lead to increasing or reducing the complexity. The strategy used in the DEC Algorithm is based on such considerations. The idea is to split the original problem in two or more sub-problems, based on precedence constraints, solve each sub-problem separately and then recombine the solutions. Ideally, the target is to find k subsets of vertices such that a solution to the original instance is the concatenation of the solutions to these k sub-problems. A decomposition is particularly interesting when the subproblems are of the same type as the parent, thus making it possible to use the same algorithm and apply it recursively. An example of a favorable situation is given in Figure 1(a) with k = 2. Vertex 5 is connected to all other vertices; so, it is known in advance which vertices will precede or follow Vertex 5. Such a vertex will be referred to as a fixed vertex. Following the scheme in Figure 1(b), each feasible solution starts with Vertex 1 followed by a permutation of Vertices $\{2, 7\}$, followed by the fixed vertex 5, followed by a permutation of Vertices $\{3, 6, 4\}$, and the sequence ends with Vertex 8. In this case, we can split the problem into two parts: the first part consists of Vertices $\{1, 2, 7, 5\}$ and the second part consists of Vertices $\{5, 3, 6, 4, 8\}$. The idea is integrated in a branch-and-bound framework, where branching is achieved by imposing an artificial precedence constraint on a selected arc in the first new branch-and-bound node, and

Vassilis Papapanagiotou was supported by the Swiss National Science Foundation (project 200020-134675/1).

Ghassan Shobaki also contributed to this work while he was at Princess Sumaya University for Technology, Amman, Jordan

(a) Precedence relation involving the fixed node 5. The dashed arrows represent a subset of the precedence relations between the nodes.

(b) Structure of each feasible solution. The two subproblems are needed to find the optimal permutation of the nodes in the two boxes. Fixed node 5 becomes the end node and the start node of the first and the second subproblem, respectively.

Figure 1: Fixed node decomposition

its opposite in the second one. The aim is to create new artificial fixed vertices in the different branches of the search-tree. The subproblems resulting from the fixed node decomposition are then represented as mixed integer linear programs and solved to optimality when small enough. The details of the DEC method can be found in Montemanni et al. [20].

THE BRANCH AND BOUND APPROACH (B&B)

Branch-and-bound enumeration is a systematic way of enumerating all possible solutions (the solution space). The solution space may be represented by a rooted tree, in which the root node is an empty solution, inner nodes are partial solutions, and each leaf node is a complete solution. Prior to augmenting a partial solution at a given tree node by adding one more vertex to it, lower bounds are computed on the optimal solution that may be found below that tree node. If the computed lower bound is not less than the best solution found, the sub-tree below that current node is discarded (pruned), because it cannot have a better solution than the best solution found.

Fast enumeration requires efficient pruning techniques that minimize the number of explicitly enumerated tree nodes. In the B&B algorithm evaluated in this paper, the pruning techniques applied at each tree node are edge-based lower bounds and history utilization. These pruning techniques are described in detail in the original paper [21] and are only summarized here for a quick reference. The idea of the edgebased lower bounds is that in a Hamiltonian path each vertex other than the start and end vertices will have exactly one incoming edge and one outgoing edge. Therefore, a lower bound on the path cost may be computed by taking the minimum-cost outgoing (incoming) edge for each vertex and summing the costs of these edges. The idea of the history utilization technique is to store solutions to previously visited sub-problems and use these solutions to quickly process related sub-problems that may be encountered later.

In addition to the two pruning techniques described in the above paragraph, a new lower bound technique based on minimum-cost bipartite matching (MCBM) is used in this work. This lower bound technique is an enhancement of the edge-based lower bound, in which the problem of simultaneously finding outgoing and incoming edges is formulated as a MCBM problem. An optimal solution to the MCBM problem gives a tighter lower bound on the optimal path cost.

COMPUTATIONAL EXPERIMENTS

Three benchmark libraries that have used in published work are considered in this evaluation: TSPLIB [22], SOPLIB06 [12] and COMPILERS [21]. These benchmarks originate from different domains and have different characteristics, thus making the comparison of the two exact algorithms more interesting. All experiments were carried out on a Quad-Core AMD Opteron 2350 processor

running at 2.0 GHz with 32GB of RAM. Only one core was used in each run. A computation time limit of 48 hours per instance was used in all experiments. For the DEC method the MILP solver adopted in the experiments is IBMs ILOG CPLEX 12.6 (http://www.cplex.com). The experimental results are shown in Tables 1, 2 and 3 (one table for each benchmark set). For each instance we report the density of the transitive closure of the precedence graph (calculated as $\frac{2 \cdot |R|}{|V| \cdot (|V|-1)}$, the best known lower bound (LB) and upper bound (UB) (according to published results), and the results produced by each of two algorithms under study (DEC and B&B). For each algorithm, we report the LB and UB obtained, the size of the optimality gap (calculated as (UB-LB)/UB) and the time in seconds required to prove optimality (a dash is reported if the algorithm does not solve an instance to optimality). An instance name in *italic* indicates that the instance has not been solved to optimality yet. An instance name in **boldface** highlights an instance that has been closed for the first time in this work. Similarly, lower or upper bounds in **boldface** indicate new best known bounds found in this work. In the last row of each table, the average optimality gap is reported for each algorithm.

The TSPLIB instances in Table 1 are the most studied instances, and the quality of the bounds available is already extremely high. The DEC algorithm was able to solve to optimality 14 of the 41 instances, while the B&B algorithm closed 10 instances. The average optimality gaps are 6.4% for DEC and 34% for B&B. On these instances, DEC appears to perform better. Examining the time taken by each algorithm to close an instance leads to an interesting observation. While DEC needs a relatively long time to close some instances, B&B tends to prove optimality in a shorter time (on average, shorter than DEC) or to spend the available time without substantial improvements. It can be said that DEC exhibits a constant but slow rate of improvement of the bounds.

For approximately half of the SOPLIB06 instances in Table 2, no optimality proof has been provided yet. The DEC algorithm was able to close 18 instances, while B&B closed 21. The evolution of bounds over time for the two algorithms seems to follow the same pattern that was observed on TSPLIB. While DEC reaches optimality near the end of the computation time limit on some instances, B&B is typically either fast in closing an instance or fails to prove optimality within the time limit. This characteristic is reflected in the optimality gaps, which are 22.7B&B. A number of new improved bounds have been found: 9 lower bounds and 1 upper bound for B&B and 7 lower bounds for DEC. These improvements led to optimality proofs for 9 new instances: 7 by B&B and 2 by both methods. Overall, it may be stated that B&B performs better than DEC on this benchmark set.

The results for the COMPILERS set are reported in Table 3. Optimality is known for all but one of the 27 instances in this set [21]. The B&B algorithm that was originally developed to solve instances in the compilers domain is performing better than DEC on this set. The number of closed instances is 26 for B&B and only 16 for DEC, with an average gap of 1.8% for B&B and 3.4% for DEC[1]. It is interesting to observe that for these instances the bound improvement over time appears to follow a different pattern to those observed for TSPLIB and SOPLIB06 instances. DEC, apart from one case, tends to either close an instance quickly or fail. Finally, it is interesting to observe that DEC computed improved lower and upper bounds for the open instance.

A closer look at the results leads to more general conclusions about the relative performance of the two algorithms. Although it is difficult to predict how successful an algorithm will be on a given instance, it is possible to observe that DEC usually performs well on instances with extreme densities of precedence constraints (either very low or a very high) but does not perform very well on instances with medium densities. On the other hand, B&B appears to be less effective

[1] Some preliminary tests mentioned in [21] were indicating that DEC was performing more poorly on these instances. The use of the transitive closure of the precedence graph led to the current improved results.

on instances with a low density of precedence constraints but is more effective than DEC on instances with medium densities. It is also interesting to note that B&B performs very well on the COMPILERS instances that are characterized by symmetrical cost graphs (the cost from vertex i to vertex j is always equal to the cost from vertex j to vertex i for all COMPILERS instances). The success of B&B on such instances is attributed to the effectiveness of the history utilization pruning technique on the more symmetrical cases. When edge costs are symmetrical, it will be more likely for the current sub-problem to relate to a previously visited subproblem, thus enabling more history-based pruning.

Table 1: TABLE 1: TSPLIB INSTANCES

Instances	Density of P	Best Known		DEC				B&B			
		LB	UB	LB	UB	Gap	Sec	LB	UB	Gap	Sec
br17.10	0.314	55	55	55	55	0.000	344.37	55	55	0.000	1.29
br17.12	0.359	55	55	55	55	0.000	491.53	55	55	0.000	181.02
ESC07	0.611	2125	2125	2125	2125	0.000	0.03	2125	2125	0.000	0.02
ESC11	0.359	2075	2075	2075	2075	0.000	0.12	2075	2075	0.000	0.28
ESC12	0.396	1675	1675	1675	1675	0.000	0.67	1675	1675	0.000	0.25
ESC25	0.177	1681	1681	1681	1681	0.000	23.48	1681	1681	0.000	46.13
ESC47	0.108	1288	1288	1288	1288	0.000	8168.49	917	3843	0.761	-
ESC63	0.173	62	62	62	62	0.000	79.63	55	76	0.276	-
ESC78	0.139	18230	18230	18230	18230	0.000	3216.30	9360	22600	0.586	-
ft53.1	0.082	7531	7531	7118	7612	0.065	-	5806	10404	0.442	-
ft53.2	0.094	8026	8026	7290	8113	0.101	-	5806	12175	0.523	-
ft53.3	0.225	10262	10262	8760	10310	0.150	-	5818	13435	0.567	-
ft53.4	0.604	14425	14425	13665	14425	0.053	-	14425	14425	0.000	4988.76
ft70.1	0.063	39313	39313	39033	39514	0.012	-	37603	46060	0.184	-
ft70.2	0.075	40101	40419	39372	40976	0.039	-	37667	48359	0.221	-
ft70.3	0.142	42535	42535	41104	42687	0.037	-	38320	52067	0.264	-
ft70.4	0.589	53530	53530	50527	53629	0.058	-	42193	54645	0.228	-
rbg048a	0.444	351	351	351	351	0.000	45.74	327	438	0.253	-
rbg050c	0.459	467	467	467	467	0.000	123.12	436	568	0.232	-
rbg109a	0.909	1038	1038	1038	1038	0.000	21910.10	1038	1038	0.000	2669.70
rbg150a	0.927	1750	1750	1747	1750	0.002	2446.44	1629	1999	0.185	-
rbg174a	0.929	2033	2033	2030	2033	0.001	-	1892	2444	0.226	-
rbg253a	0.948	2950	2950	2928	2950	0.007	-	2754	3558	0.226	-
rbg323a	0.928	3140	3140	3131	3146	0.005	-	2933	4032	0.273	-
rbg341a	0.937	2568	2568	2470	2626	0.059	-	2153	3786	0.431	-
rbg358a	0.886	2545	2545	2479	2654	0.066	-	2232	4110	0.457	-
rbg378a	0.894	2809	2816	2693	2922	0.078	-	2260	4109	0.450	-
kro124p.1	0.046	38762	39420	36551	41177	0.112	-	33498	52575	0.363	-
kro124p.2	0.053	39841	41336	36966	43311	0.146	-	33787	57723	0.415	-
kro124p.3	0.092	43904	49499	37814	53016	0.287	-	33872	68962	0.509	-
kro124p.4	0.496	73021	76103	55876	77139	0.276	-	39983	92438	0.567	-
p43.1	0.101	28140	28140	28083	28140	0.002	-	690	28480	0.976	-
p43.2	0.126	28480	28480	28124	28480	0.013	-	690	28795	0.976	-
p43.3	0.191	28835	28835	27309	28835	0.053	-	690	29300	0.976	-
p43.4	0.614	83005	83005	82933	83005	0.001	-	83005	83005	0.000	938.20
prob.100	0.048	1045	1163	1000	1973	0.493	-	761	2071	0.633	-
prob.42	0.116	243	243	243	243	0.000	32404.10	159	329	0.517	-
ry48p.1	0.091	15805	15805	14836	15805	0.061	-	12216	20190	0.395	-
ry48p.2	0.103	16074	16666	14974	16741	0.106	-	12216	19424	0.371	-
ry48p.3	0.193	19490	19894	16290	19894	0.181	-	12528	22483	0.443	-
ry48p.4	0.588	31446	31446	27061	31446	0.139	-	31446	31446	0.000	2471.05
Average Gap						0.064				0.340	

Table 2: TABLE 2: SOPLIB06 INSTANCES

Instances	Density of P	Best Known		DEC				B&B			
		LB	UB	LB	UB	Gap	Sec	LB	UB	Gap	Sec
R.200.100.1	0.020	61	61	61	61	0.000	3786.22	61	400	0.848	-
R.200.100.15	0.847	1257	1792	1243	2225	0.441	-	810	4074	0.801	-
R.200.100.30	0.957	4185	4216	4216	4216	0.000	86869.60	4216	4216	0.000	741.45
R.200.100.60	0.991	71749	71749	71749	71749	0.000	0.15	71749	71749	0.000	32.13
R.200.1000.1	0.020	1404	1404	1403	1404	0.001	-	1392	5022	0.723	-
R.200.1000.15	0.867	14565	20481	14689	24527	0.401	-	10104	40333	0.749	-
R.200.1000.30	0.958	40170	41196	41196	41196	0.000	170875.00	41196	41196	0.000	517.29
R.200.1000.60	0.989	71556	71556	71556	71556	0.000	0.15	71556	71556	0.000	35.29
R.300.100.1	0.013	26	26	26	104	0.750	-	26	363	0.928	-
R.300.100.15	0.905	2166	3161	2160	3796	0.431	-	1298	6640	0.805	-
R.300.100.30	0.970	5839	6120	5846	6133	0.047	-	6120	6120	0.000	3364.72
R.300.100.60	0.994	9726	9726	9726	9726	0.000	0.18	9726	9726	0.000	141.47
R.300.1000.1	0.013	1294	1294	1292	2011	0.358	-	1291	6078	0.788	-
R.300.1000.15	0.905	21096	29183	21077	36766	0.427	-	14352	57296	0.750	-
R.300.1000.30	0.965	51495	54147	51789	54744	0.054	-	54147	54147	0.000	8390.30
R.300.1000.60	0.994	109471	109471	109471	109471	0.000	0.32	109471	109471	0.000	176.08
R.400.100.1	0.010	13	13	13	90	0.856	-	13	285	0.954	-
R.400.100.15	0.927	2747	3906	2737	5115	0.465	-	1811	8794	0.794	-
R.400.100.30	0.978	7755	8165	8055	8167	0.014	-	8165	8165	0.000	65486.19
R.400.100.60	0.996	15228	15228	15228	15228	0.000	1.97	15228	15228	0.000	491.28
R.400.1000.1	0.010	1343	1343	1343	2032	0.339	-	1336	4437	0.699	-
R.400.1000.15	0.930	28159	29685	28207	49825	0.434	-	19886	81262	0.755	-
R.400.1000.30	0.977	79868	85132	82264	85223	0.035	-	85128	85128	0.000	2699.44
R.400.1000.60	0.995	140816	140816	140816	140816	0.000	0.51	140816	140816	0.000	494.20
R.500.100.1	0.008	4	4	4	85	0.953	-	4	383	0.990	-
R.500.100.15	0.945	3543	5361	3523	6813	0.483	-	2370	11486	0.794	-
R.500.100.30	0.980	8600	9665	8693	10026	0.133	-	9665	9665	0.000	21075.92
R.500.100.60	0.996	18240	18240	18240	18240	0.000	0.50	18240	18240	0.000	1301.61
R.500.1000.1	0.008	1316	1316	1315	2090	0.371	-	1313	6205	0.788	-
R.500.1000.15	0.940	32950	50725	32522	64092	0.493	-	22597	111129	0.797	-
R.500.1000.30	0.981	91272	98987	93698	100270	0.066	-	98987	98987	0.000	26041.65
R.500.1000.60	0.996	178212	178212	178212	178212	0.000	0.57	178212	178212	0.000	1466.67
R.600.100.1	0.007	1	1	1	85	0.988	-	1	378	0.997	-
R.600.100.15	0.950	3656	5684	3611	7479	0.517	-	2355	13271	0.823	-
R.600.100.30	0.985	11841	12465	12127	12527	0.032	-	12465	12465	0.000	39004.34
R.600.100.60	0.997	23293	23293	23293	23293	0.000	0.48	23293	23293	0.000	1945.58
R.600.1000.1	0.007	1337	1337	1337	1337	0.000	131432.00	1336	4931	0.729	-
R.600.1000.15	0.945	36546	57237	36575	73389	0.502	-	27096	120975	0.776	-
R.600.1000.30	0.984	116037	126789	118311	129869	0.089	-	79564	178608	0.555	-
R.600.1000.60	0.997	214608	214608	214608	214608	0.000	0.67	214608	214608	0.000	3652.19
R.700.100.1	0.006	1	1	1	1	0.000	13781.60	1	446	0.998	-
R.700.100.15	0.957	4494	7311	4499	9213	0.512	-	3117	14643	0.787	-
R.700.100.30	0.987	13663	14510	14084	14601	0.035	-	8994	19138	0.530	-
R.700.100.60	0.997	24102	24102	24102	24102	0.000	1.49	24102	24102	0.000	6561.80
R.700.1000.1	0.006	1231	1231	1231	1231	0.000	56712.00	1228	4886	0.749	-
R.700.1000.15	0.956	40662	66837	40808	88621	0.540	-	29226	151331	0.807	-
R.700.1000.30	0.986	118718	134474	121173	138442	0.125	-	80969	187072	0.567	-
R.700.1000.60	0.997	245589	245589	245589	245589	0.000	1.54	245589	245589	0.000	8243.04
Average Gap						0.227				0.443	

Table 3: TABLE 3: COMPILERS INSTANCES

Instances	Density of P	Best Known		DEC				B&B			
		LB	UB	LB	UB	Gap	Sec	LB	UB	Gap	Sec
gsm.153.124	0.970	1109	1109	1109	1109	0.000	972.61	1109	1109	0.000	1.82
gsm.444.350	0.990	2745	2745	2745	2745	0.000	1.29	2745	2745	0.000	1.58
gsm.462.77	0.840	577	577	577	577	0.000	3.77	577	577	0.000	17.47
jpeg.1483.25	0.484	93	93	93	93	0.000	9.47	93	93	0.000	6.26
jpeg.3184.107	0.887	769	793	769	793	0.030	-	791	791	0.000	82.19
jpeg.3195.85	0.740	28	68	28	68	0.588	-	68	68	0.000	47.66
jpeg.3198.93	0.752	304	312	304	312	0.026	-	312	312	0.000	53.20
jpeg.3203.135	0.897	830	852	830	852	0.026	-	850	850	0.000	306.84
jpeg.3740.15	0.257	40	40	40	40	0.000	4.12	40	40	0.000	3.12
jpeg.4154.36	0.633	167	167	167	167	0.000	288.28	167	167	0.000	18.55
jpeg.4753.54	0.769	241	245	241	245	0.016	-	245	245	0.000	25.00
susan.248.197	0.939	1338	1338	1304	1338	0.025	-	1338	1338	0.000	672.84
susan.260.158	0.916	1016	1016	945	1016	0.070	-	1016	1016	0.000	81.91
susan.343.182	0.936	1207	1207	1159	1207	0.040	-	1207	1207	0.000	195.43
typeset.10192.123	0.744	214	630	565	602	0.061	-	328	634	0.483	-
typeset.10835.26	0.349	127	127	127	127	0.000	114.55	127	127	0.000	55.74
typeset.12395.43	0.518	174	174	168	174	0.034	-	174	174	0.000	527.88
typeset.15087.23	0.557	98	98	98	98	0.000	0.33	98	98	0.000	10.01
typeset.15577.36	0.555	155	155	154	155	0.006	-	155	155	0.000	114.64
typeset.16000.68	0.658	84	84	84	84	0.000	6624.72	84	84	0.000	52.91
typeset.1723.25	0.245	64	64	64	64	0.000	40.32	64	64	0.000	40.38
typeset.19972.246	0.993	2018	2018	2018	2018	0.000	2.63	2018	2018	0.000	1.84
typeset.4391.240	0.981	1605	1605	1605	1605	0.000	4815.67	1605	1605	0.000	40.29
typeset.4597.45	0.493	184	184	184	184	0.000	22868.00	184	184	0.000	1017.70
typeset.4724.433	0.995	3466	3466	3466	3466	0.000	46.66	3466	3466	0.000	11.22
typeset.5797.33	0.748	131	131	131	131	0.000	0.48	131	131	0.000	69.50
typeset.5881.246	0.986	1732	1732	1732	1732	0.000	686.96	1732	1732	0.000	47.73
Average Gap						0.034				0.018	

References

[1] L.F. Escudero, *"An inexact algorithm for the sequential ordering problem"*, European Journal of Operational Research, vol. 37, no. 2, pp. 236-249, 1988.

[2] N. Ascheuer, L.F. Escudero, M. Groetschel and M. Stoer, *"A cutting plane approach to the sequential ordering problem (with applications to job scheduling in manufacturing)"*, SIAM Journal on Optimization, vol. 3, no.1, pp. 2542, 1993.

[3] L.F. Escudero, M. Guignard and K. Malik, *"A Lagrangian relax-and-cut approach for the sequential ordering problem with precedence relationships"*, Annals of Operations Research, vol. 50, no.1, pp. 219237, 1994.

[4] I.T. Hernadvolgyi, *"Solving the sequential ordering problem with automatically generated lower bounds"*, In Proceedings of Operations Research, Heidelberg, Springer Verlag, pp. 355-362, 2003.

[5] I.T. Hernadvolgyi, *"Automatically generated lower bounds for search"*, PhD thesis, School of Information Technology and Engineering, University of Ottawa, Canada, 2004.

[6] E. Balas, M. Fischetti and W.R. Pulleyblank, *"The precedence-constrained asymmetric traveling salesman polytope"*, Mathematical Programming, vol. 68 no. 1, pp. 241265, 1995.

[7] S. Chen and S. Smith, *"Commonality and genetic algorithms"*, Technical Report CMU-RI-TR-96-27, Robotics Institute, Carnegie Mellon University, Pittsburgh, USA, 1996.

[8] C. Moon, J. Kim, G. Choi and Y. Seo, *"An efficient genetic algorithm for the traveling salesman problem with precedence constraints"*, European Journal of Operational Research, vol. 140, no. 3, pp. 606617, 2002.

[9] W.R. Pulleyblank and M. Timlin, *"Precedence constrained routing and helicopter scheduling: Heuristic design"*, Technical Report RC17154 (#76032), IBM T.J. Watson Research Center, New York, USA, 1991.

[10] L.M. Gambardella and M. Dorigo, *"An ant colony system hybridized with a new local search for the sequential ordering problem"*, INFORMS Journal on Computing, vol. 12, no. 3, pp. 237255, 2000.

[11] R. Montemanni, D.H. Smith and L.M. Gambardella, *"Ant colony systems for large sequential ordering problems"*, In Proceedings of the IEEE Swarm Intelligence Symposium (SIS), pp. 6067, Honolulu, USA, 2007.

[12] R. Montemanni, D.H. Smith and L.M. Gambardella LM, *"A heuristic manipulation technique for the sequential ordering problem"*, Computers and Operations Research, vol. 35, no.12, pp. 39313944, 2008.

[13] R. Montemanni, D.H. Smith, A.E. Rizzoli and L.M. Gambardella, *"Sequential ordering problems for crane scheduling in port terminals"*, In Bruzzone et al., editors, Proceedings of HMS - The 11th Intermodal Workshop on Harbor, Maritime and Multimodal Logistic Modeling and Simulation, pp. 180189, Campora San Giovanni, Italy, 2008.

[14] R. Montemanni, D.H. Smith, A.E. Rizzoli and L.M. Gambardella, *"Sequential ordering problems for crane scheduling in port terminals"*, International Journal of Simulation and Process Modelling, vol. 5, no. 4, pp. 348361, 2009.

[15] D. Anghinolfi, R. Montemanni, M. Paolucci and L.M. Gambardella, *"A particle swarm optimization approach for the sequential ordering problem"*, , In Proceedings of the VIII Metaheuristic International Conference (MIC), Hamburg, Germany, 2009.

[16] D. Anghinolfi, R. Montemanni, M. Paolucci and L.M. Gambardella, *"A hybrid particle swarm optimization approach for the sequential ordering problem"*, Computers & Operations Research, vol. 38, no. 7, pp. 10761085, 2011.

[17] M. Mojana, R. Montemanni, G.A. Di Caro and L.M. Gambardella, *"An algorithm combining linear programming and an ant system for the sequential ordering problem"*, In Proceedings of ATAI - The Second Annual International Conference on Advanced Topics in Artificial Intelligence, pp. 8085, Singapore, 2011.

[18] M. Mojanaa, *"Matheuristic techniques for the sequential ordering problem"*, Master thesis, Faculty of Informatics, Universit della Svizzera italiana, Lugano, Switzerland, 2011.

[19] M. Mojana, R. Montemanni, G.A. Di Caro and L.M. Gambardella, *"A branch and bound approach for the sequential oridering problem"*, In P. Luangpaiboon et al. eds., editor, Lecture Notes in Management Science, vol. 4, pp. 266273, 2012.

[20] R. Montemanni, M. Mojana, G.A. Di Caro and L.M. Gambardella, *"A Decomposition-based exact approach for the sequential ordering problem"*, Journal of Applied Operational Research, vol. 5, no. 1, pp. 213, 2013.

[21] G. Shobaki and J. Jamal, *"An exact algorithm for the sequential ordering problem and its application to switching energy minimization in compilers"*, Computational Optimization and Applications, vol. 61, no. 2, pp. 343-372, 2015.

[22] N. Ascheuer, *"Hamiltonian path problems in the on-line optimization of flexible manufacturing systems"*, Technical Report 3, Konrad-Zuse-Zentrum fur Informationstechnik Berlin, Heilbronner Str. 10, D-10711 Berlin-Wilmersdorf, 1996.