An integrated solver for multi-index assignment

Stathis Plitsos
Dep. of Management Sci. & Technology
Athens University of
Economics and Business,
Athens, Greece
stathisp@aueb.gr

Dimitrios Magos
Dep. of Informatics,
Technological Educational
Institute of Athens,
Athens, Greece
dmagos@teiath.gr

Ioannis Mourtos
Dep. of Management Sci. & Technology
Athens University of
Economics and Business,
Athens, Greece
mourtos@aueb.gr

Abstract

We study the axial and planar multi-index assignment problem under the lens of the (k,s)-Assignment problem, i.e., a framework that encompasses all assignment structures. Taking advantage of the complementary strengths of approaches to optimization, we develop for the problem at hand an integrated solver that offers, constraint propagation, problem-specific cuts, SoS-I branching and a new variant of the Feasibility Pump heuristic that uses cutting planes within the pumping cycles of the algorithm. All these components, when combined in a B&C algorithm for the 3-index axial and planar cases, reduce both the time and the number of nodes in the search tree, particularly for larger instances. This effect is more evident in the planar case, where the number of constraints increases quadratically compared to the axial assignment. Apart from literature, non-polynomial, instances we generate and examine large-size instances for which a competitive commercial solver runs out of memory. Further research includes testing this algorithm on the 4-etc-index size axial and planar assignment problems.

Introduction

Assignment structures are embedded in many optimization problems. In general, an assignment occurs whenever a member of an entity must be allocated to a member of another entity. The simplest case is the well-known 2-index assignment (Martello and Toth 1987) equivalent to the weighted bipartite matching. Other well-studied variants of assignments are the 3-index axial (Pierskalla 1968) and planar (Frieze 1983) assignments among with their generalization, i.e., the multi-index axial and planar assignment problems. There are numerous applications of these assignment problems. For example, axial assignments apply to data-association problems (Poore and Gadaleta 2006), wafer-to-wafer yield optimization in 3D electronic circuit printing (Taouil and Hamdioui 2011), statistical design of experiments (Higgins 2013), while planar assignments share the diverse applications of orthogonal and mutually orthogonal Latin squares (Laywine and Mullen 1998).

The $(k,s)-$Assignment Problem (Appa, Magos, and Mourtos 2006) encompasses all these assignment structures. Hereafter we denote it as $(k,s)AP_n$. This problem involves the elements of k disjoint n-sets such that each s-tuple of elements, each from a different set, appears exactly once at any given solution. Parameter k defines the number of sets, n determines the cardinality of the sets while s determines the type of assignment. Hence, its Integer Programming (IP) model consists of n^k, k-indexed binary variables and $\binom{k}{s}\cdot n^s$ equality constraints each of which has s indices fixed to specific values and $k-s$ indices to be summed over all the values of their domains. The right-hand side for each constraint is equal to one.

To formally define this IP, let $K=\{1,\ldots,k\}$ and $S\subseteq K$ denote a subset of s indices (out of k). Let $Q_{k,s}$ denote the collection of all such distinct S, i.e., $Q_{k,s}=\{S\subseteq K : |S|=s\}$ and $|Q_{k,s}|=\binom{k}{s}$. Consider k disjoint n -sets $M_1,M_2,\ldots,M_k$ and let $m^i\in M_i,\forall i\in K$. For $S\subseteq K$, assume $S=\{i_1,i_2,\ldots,i_s\}$ such that $i_1<i_2<\cdots<i_s$. Let $M^S=M_{i_1}\times M_{i_2}\times\cdots\times M_{i_s}$ and $m^S\in M^S$, where $m^S=(m^{i_1},m^{i_2},\ldots,m^{i_s})$. Observe that, for $S'\subset S\subseteq K$, $m^{S'}$ defines a subset of indices of the tuple m^S, i.e. $m^{S'}\subset m^S$. Further, we follow the convention that $m_0^i,m_1^i,\ldots$ denote the elements of the set M_i, i.e. the values of the index m^i. In an analogous manner, m_0^S denotes the specific s-tuple $(m_0^{i_1},\ldots,m_0^{i_s})$. For the mathematical programming formulation of $(k,s)AP_n$, consider binary variables x_{m^K} and the mapping $w: M^K\rightarrow R$. The problem is formulated as follows (Appa, Magos, and Mourtos 2006).

$$\min\sum\{w_{m^K}\cdot x_{m^K} : m^K\in M^K\},$$
$$\sum\{x_{m^K} : m^{K\setminus S}\in M^{K\setminus S}\}=1,\forall m^S\in M^S, S\in Q_{k,s},$$
$$x_{m^K}\in\{0,1\}^{n^k},\forall m^K\in M^K.$$

Taking advantage of the complementary strengths of different approaches to optimization (Hooker 2012), we develop under the lens of the $(k,s)AP_n$ a solver integrating several algorithmic components, namely, constraint propagation, problem-specific cuts, SoS-I branching among with a new variant of the Feasibility Pump (FP) heuristic that employees both constraint propagation and cuts to obtain a better feasible solution. Having such a tool-set available, we focus on the 3-index axial ($(3,1)AP_n$) and planar assignment ($(3,2)AP_n$) and present a Branch and Cut (B&C) algorithm, that employees all these components. We test all these components of the B&C method, among with the method itself, on the small and medium size non-polynomial literature instances and also on larger instances generated for the purposes of this work.

The results of this experimentation show that indeed the proposed FP variant produces better feasible solutions, while in cooperation with SoS-I branching and problem-specific cuts, reduces significantly the time and the number of nodes in the search tree, especially for large-size instances. Hence, the contribution of this work is threefold:

- we investigate the effect of problem-specific cutting planes on the performance of the FP heuristic;
- we present a B&C for the 3-index axial and planar assignment problems which, to the best of our knowledge, is missing from the (rather extensive) literature;
- we present an integrated solver, in terms of code, that can be tested on any k-index axial and planar assignment problems.

Algorithmic components

The algorithmic components developed and used include problem-specific cuts, constraint propagation and a variant of the FP heuristic that employees both constraint propagation and cuts to obtain a better feasible solution. Let us describe each component with the respective research background individually.

Cutting planes

For both planar and axial assignments we use problem-specific cuts as obtained from the literature. Regarding the axial assignment, we employ the two known families of facet-defining inequalities arising from cliques of the intersection graph associated with the formulation of the $(k, 1)AP_n$ (Magos and Mourtos 2009), which are polytime separable; for $k = 3$, these are the only families of clique inequalities (Balas and Saltzman 1989) for which the separation algorithm used is proven to be $O(n^3)$, i.e, of linear time with respect to the number of variables (Balas and Qi 1993).

For the planar assignment, we use odd-holes cuts, again induced from the intersection graph, as described in the literature (Padberg 1973). The separation algorithm used has a complexity of $O(n^8)$ (Magos and Mourtos 2013). Clique-cuts cannot be used on the 3-index planar assignment problem, since the only cliques arising from the intersection graph are the constraints of the problem itself (Euler, Burkard, and Grommes 1986).

Constraint Propagation

We implement a mechanism that performs constraint propagation for all classes of $(k, s)AP_n$ (i.e., for all values of k, s and n) and capable of supporting any heuristic. This mechanism includes an oracle returning which variables participate in which constraint; two propagating functions $setToOne$ and $setToZero$ and a backtracking function, invoked when infeasibility is detected.

We use a stack of size n^k, i.e., equal to the number of binary variables, to store the variables that are set to a value together with a flag per variable indicating whether the variable pushed is *set-by-choice* or *set-by-force* (i.e., forced to a value by constraint propagation). This allows backtracking to jump to the last *set-by-choice* variable and set it to its complement hence making it *set-by-force*. At any point, the stack being empty (full) implies that infeasibility is reached (a feasible solution is found).

Once a variable x is set to one, $setToOne$ calls $setToZero$ for all variables appearing in some constraint together with x. Once a variable is set to zero, $setToZero$ checks if in any constraint this variable appears, there remains a single variable not set to a value, and calls for this variable $setToOne$ (if this fails, infeasibility is detected). Using these two functions, our code implements a recursive depth-first propagation on the constraints of $(k, s)AP_n$. Since the number of these constraints is $\binom{k}{s} \cdot n^s$ (Appa, Magos, and Mourtos 2006), i.e., it increases with s, constraint propagation is more effective for planar rather than axial problems.

Feasibility Pump

Across the literature there have been several attempts to tackle heuristically the problem at hand, especially the axial case, e.g., (Crama and Spieksma 1992; Bandelt, Crama, and Spieksma 1994; Burkard, Rudolf, and Woeginger 1996). For an extensive review on such approaches on the axial case see (Karapetyan and Gutin 2011). Regarding the planar case, there has not been so many attempts (Magos 1996; Dichkovskaya and Kravtsov 2006; Gimadi and Glazkov 2007), while recent literature focuses on semi-latin and the completion of partial Latin squares, indicatively see (Euler 2010; Casselgren and Häggkvist 2013; Soicher 2013). However, all these attempts are problem-specific and cannot tackle the diversity of axial and planar assignment problems. Hence the need for a general-purpose heuristic that tackles all assignment structures.

The Feasibility Pump (FP) is a rather recent, yet extensively studied Linear Programming (LP) based heuristic for mixed integer programs. In general, this algorithm tries to minimize the distance between an LP-feasible point and its scalar rounding to the nearest integer point. Thus, the algorithm is mainly guided by feasibility. The first version of this algorithm (Fischetti, Glover, and Lodi 2005) tackles only binary problems, while the following one (Bertacco, Fischetti, and Lodi 2007) focuses on general integer ones. An additional version (Achterberg and Berthold 2007), is guided not only by feasibility, but by optimality as well, using a convex combination of the aforementioned distance and the initial cost vector. A very interesting version (Fischetti and Salvagnin 2009) uses constraint propagation in the rounding phase to obtain better feasible points, combining also the aforementioned optimality criterion. Following this, it has been shown (Boland et al. 2012) that FP can be seen as a discrete version of the proximal point algorithm, while new mechanisms for incorporating restarts of the algorithm have been suggested, such as the addition of cutting planes. Also, FP has been interpreted (De Santis, Lucidi, and Rinaldi 2013) as the Frank-Wolfe method applied to a non-smooth concave merit function, a result that leaded to new concave non-differentiable penalty functions for measuring solution integrality (De Santis, Lucidi, and Rinaldi

2014).

Intrigued by the nature of this heuristic that allows the use of different optimization methods, such as constraint propagation, we investigated the effect of problem-specific cuts before and in the pumping cycles of the algorithm. This is different from an approach that used cuts in the restart-phase (Boland et al. 2012), since we use cuts before the pumping phase to tighten the LP and get a "better" fractional point, and in the pumping phase to enforce feasibility and reduce the number of cycles. For the $(3,1)AP_n$ we add clique cuts while for the $(3,2)AP_n$ odd-hole cuts, using the separation algorithms described previously. For both cases cuts are added for up to 1 round and up to $n^s/9$ cuts per round. Considering that we treat the problem at hand as a minimization problem, adding cuts in the heuristic implies that we use them to improve the upper-bound (whereas, typically, adding cuts within B&C utilizes them for improving the lower bound).

We focus on the objective version which we denote as OFP1 (Achterberg and Berthold 2007), OFP1 with constraint propagation (Fischetti and Salvagnin 2009), hereafter denoted as OFP2, and the reweighed version, as called in the literature (De Santis, Lucidi, and Rinaldi 2014) and denoted hereafter as ORFP1. For all of these versions, we investigate the effect of cuts in the performance of these algorithms. OFP2 with cuts is denoted as OFP3, while ORFP1 with constraint propagation is ORFP2 and when adding cuts too we get ORFP3. Cuts are added in the pumping phase only while parameter $\alpha < 0.5$ (Achterberg and Berthold 2007) and $\theta < 0.5$ (De Santis, Lucidi, and Rinaldi 2013). For all of these versions, the parameters of the algorithms are the same as in the literature. Particularly for the variants that employ constraint propagation, we use our problem-specific constraint propagation mechanism, using the '*frac*' rounding scheme (Fischetti and Salvagnin 2009), i.e., less fractional variables are rounded first.

Branch & Cut algorithm

All these algorithmic components allow the development of an exact algorithm that employees the complementary strengths of these methods. Still, both of these problems, i.e., $(3,1)AP_n$ and $(3,2)AP_n$, have been studied across the literature and there exist exact approaches to tackle them. However, a B&C algorithm, to the best of our knowledge, does not exist for both of these problems. Before going into the details of our approach let us describe in short these previous approaches.

Regarding the $(3,1)AP_n$, the first exact approach is a B&B algorithm, merely a form of implicit enumeration of all feasible points, based on the primal-dual scheme for upper and lower bound acquisition (Pierskalla 1968). Following this an additional exact, primal-dual algorithm has been proposed (Hansen and Kaufman 1973). Both of the above algorithms share a branching strategy based on fixing single variables to zero or one. After that, different approaches came in the front stage where an alternation of the $(3,1)AP_n$ fitting the application of scheduling a teaching practice is solved exactly, using a Lagrangian relaxation and a subgradient algorithm (Frieze and Yadegar 1981). An interesting approach introduces a B&B algorithm that incorporates a Lagrangian relaxation for which dual heuristics have been encoded to obtain good lower bounds, facet inequalities which 'tighten' the relaxation and primal heuristics that obtain good upper bounds (Balas and Saltzman 1991). Following this, the effect of clique-cuts added only at the top-node has been investigated in a B&B algorithm (Qi, Ballas, and Gwan 1994). The latest approach (Walteros et al. 2014), focuses on the general $(k,1)AP_n$ with $k > 3$, introducing a Branch & Price algorithm applied to assignment problems with star costs, i.e., a special form of costs. Note here that we do not present a detailed literature review on this since we are focusing on the 3-index assignment.

On the contrary, the $(3,2)AP_n$ has not been so extensively studied. The first exact algorithm is a B&B algorithm (Vlach 1967). Following this, another B&B algorithm has been proposed (Magos and Miliotis 1994), where a relaxation-based heuristic and a local-improvement algorithm form the upper bound procedure, a dual heuristic solving a Lagrangian relaxation of the problem obtains lower bounds, while branching is imposed over SoS-I where variables are prioritized in accordance to their reduced costs at each node of the search tree. Again, other approaches tackle higher dimensions of planar assignments, e.g., $k = 4$ (Appa, Magos, and Mourtos 2004), however, in this study, we do not focus on these ones.

Our approach uses clique-cuts for the $(3,1)AP_n$, odd-hole cuts for the $(3,2)AP_n$, branching on special ordered sets and the reweighed FP heuristic with constraint propagation and cuts (ORFP3). Let us describe how we use these components individually.

Branching rules

As described previously, the $(k,s)AP_n$ has strictly binary variables where all constraints are equalities with a right-hand side of one. This means that each constraint defines a *Special Ordered Set of type I* (SoS-I). Hence, we perform SoS-I branching using the reduced costs of the initial LP to prioritize variables. That is, variables in each set are prioritized in increasing order with respect to their reduced costs, while each constraint is prioritized with respect to the minimum integer infeasibility (i.e., CPLEX default SoS-I branching). By branching over SoS-I, we take advantage of the $(k,s)AP_n$ structure to reduce the depth of the search tree to a maximum $O(nlogn)$ (Balas and Saltzman 1991). As an alternative prioritization criterion, regarding the variables, we have tested using the reduced costs of the variables at each node, while for constraint prioritization we used the sum of the reduced costs at each constraint. However this did not improve the computational results.

The node selection criterion is "best bound".

Cut addition

In order to design an efficient B&C scheme, it is important not only to have effective separation algorithms, but also to determine the frequency of calling them. For example, it is common to add cuts at several, but not all, nodes usually high in the search tree (Lysgaard, Letchford, and Eglese 2004; Dumitrescu et al. 2010). Therefore, we add clique cuts

for the $(3,1)AP_n$ and odd-hole cuts for the $(3,2)AP_n$ only at nodes which are at the 5% of the maximum tree depth, which is known as described previously. Our computational experience has shown that a light scheme of adding cuts in both cases, for one round and up to $n^s/9$ cuts per round, provides the best possible results. In this way we perform light bursts of cuts high in the search tree, which tighten the relaxation and reduce the search space.

Heuristic

The computational results presented below show that all variants of FP are expensive in terms of time, however when compared to each other, the variants that employ cuts with constraint propagation provide on average the best upper bounds. This implies that, indeed OFP3 and ORFP3 can prune the search tree, but calling any of these heuristics too many times in the B&C algorithm would be a burden in terms of solution time. In this study, we present results for the ORFP3 version called only once at the top-node of the tree. After that, we employ only the default heuristic of CPLEX, to further improve the upper bound.

Computational results

All variants of FP and the B&C algorithms are coded in ANSI C, using the IBM-ILOG CPLEX 12.5 callable library. Our experiments are conducted under Linux Ubuntu 14.04, on a quad-core machine (3.6GHz CPU speed, 16GB RAM). All FP variants are allowed up to 2000 pumping cycles, except when employed into a B&C algorithm where we set the maximum number of pumping cycles to 20. To investigate the effect of cuts on the performance of the FP variants we calculate the (average) integrality gap of all known FP variants, defined as $IG = (z^* - z_{LP})/z_{LP}$, where z^* is the value of the solution found by the FP variant and z_{LP} the value of the LP-relaxation. We also discuss issues related to the required CPU time. Additionally, we take into consideration the success ratio (if less than 1), i.e., the percentage of runs in which a solution is found. Finally we show the average number of pumping cycles needed by each variant to find a feasible solution.

Regarding the exact algorithms, our preliminary computational experience showed that for large instances, i.e., $n > 50$ for the axial and $n \geq 20$ for the planar case, CPLEX runs out of memory way before reaching the optimal solution. For this reason and in order to have a sound basis of comparison, we set a time limit of 3 hours on all exact schemes and calculate the integrality gap that is reached within this time-frame. To test the effect of each algorithmic component, i.e., SoS-I branching, cuts and ORFP3, we show results of 3 exact algorithms compared to the CPLEX default one, where each component is employed hierarchically. That is, *SoS-I* scheme uses only the respective branching strategy, *SoSCuts* uses SoS-I branching and cuts, and *SoSCutsORFP3* employees all three components. For these 3 schemes all general-purpose CPLEX cuts are turned off. Finally, we turn off the CPLEX pre-solver, we use explicitly the single-threaded mode, and keep the CPLEX heuristic on default settings on all exact schemes. Let us describe the results of each problem individually.

3-index axial assignment

For the $(3,1)AP_n$ we focus on two classes of non-polynomial instances found in the literature, which we denote as *bsx* (Balas and Saltzman 1991) and *gpx* (Grundel and Pardalos 2005), with x being the size of the instance, i.e., the value of n. Additionally, we create a new class of instances, denoted as axial$x$, whose coefficients are integer numbers sampled from $U[1, n^k]$, on which we test our algorithms as well. For all of these classes, we generate 5 different objective functions with $n \in \{25, 54, 66, 80\}$. Therefore, all results shown are averages over 5 random instantiations per instance.

Table 1 shows the results of the FP variants when tested on these instances. In general, OFP1 is the fastest variant, but sometimes fails to find a feasible solution. When constraint propagation and cuts are applied to it (OFP3), on average the quality of solution gets better while feasibility is reached on all instances and the number of pumping cycles is also reduced. However, as expected, this comes at the expense of time. ORFP1 is comparable to OFP1, however constraint propagation and cuts (ORFP3) indeed improve the solution quality and reduce the number of pumping cycles in the majority of the instances, but the computational time remains comparable to ORFP1. Note here that, on average, ORFP3 provides the best solution quality followed by OFP3. Therefore, cut addition in each pumping cycle is clearly beneficial although expensive.

Table 2 shows the results of the exact algorithms. When looking at the performance of CPLEX default B&C, one can easily notice the differentiation of number in nodes and solution time between different classes of instances. When looking at the literature instances,i.e., bsx and gpx, it is obvious that they are solved significantly faster compared to *axial*x instances. This is more evident in instances with $n > 25$. A recent study on heuristic approaches for the $(k,1)AP_n$ (Karapetyan and Gutin 2011), highlights that when solving a randomly generated instance with cost coefficients sampled in a range $[a, b]$, as n increases, the problem solution approaches the bound an, i.e., the minimal possible assignment weight. Given that all literature instances are generated randomly within a constant range of cost coefficients, indeed we came across this finding when solving these instances. However, in our instances (*axialx*), the range widens as n increases, making this phenomenon rarely evident and these instances 'harder' to solve as proven by the performance of the CPLEX B&C algorithm.

When looking on the bsx and gpx instances (Table 2), our exact algorithms perform comparably to CPLEX, in terms of nodes. However, this does not apply in terms of time, that is our schemes are time-wise more expensive due to the computational time required by ORPF3. This means that CPLEX suffices for these instances, mostly because of the problem particularity described above. However, when looking on the *axial*x instances, we improve on CPLEX in terms of nodes, time and integrality gap, which indicates that our B&C algorithms indeed prune the search tree more effectively. When comparing our B&C algorithms with each other on the *axial*x instances it seems that sometimes ORFP3 reduces the number of nodes but increases the time

to optimality. However, this burden, in terms of time, seems to pay off for larger instances ($n = 66$ and $n = 80$) given the integrality gap that is reached within the time-frame of 3 hours. It is expected that this becomes more significant for even larger instance that are currently examined.

3-index planar assignment

For the $(3, 2)AP_n$ we focus on a single class of literature instances (Magos and Mourtos 2013) where the cost coefficients are integer numbers sampled from $U[1, n^k]$. For this class, we generate 5 different objective functions with $n \in \{10, 20, 30, 40\}$. Therefore, all results shown are averages over 5 random instantiations per instance, while the instances are identified by the value of n.

Table 3 shows the results of the FP variants when tested on the planar instances. Again, OFP1 requires less time than the other variants, however it provides the poorest quality of solutions on average. When constraint propagation and cuts are employed (OFP3) the number of pumping cycles is reduced and the solution quality improved. Regarding ORFP1, it provides solutions of better quality than OFP1 in a comparable amount of time. When constraint propagation and cuts are employed (ORFP3) the number of pumping cycles is reduced and the solution quality is improved. However, time-wise ORFP3 is far more expensive. This is obviously due to the cut addition; recall that the separation of odd-hole cuts is $O(n^8)$.

Table 4 shows the results of the exact algorithms for the planar instances. For small instances ($n = 10$), all of our algorithmic schemes seem to perform better than CPLEX in terms of nodes and time. Particularly, the *SoSCutsORFP3* scheme requires more time, which is reasonable given the time required by ORFP3. The effect though of ORFP3 is far more evident on larger instances ($n \geq 20$). The number of nodes visited by the schemes that employ cuts is much smaller due to the time required by the separation algorithm, i.e., CPLEX and *SoS-I* scheme spend less time at each node. However the integrality gap reached by *SoS-CutsORFP3* scheme is the minimum possible, which indicates that indeed all these components prune significantly the search tree.

Concluding Remarks

In this study we present a solver for the $(k, s)AP_n$ that integrates constraint propagation, problem-specific cuts, SoS-I branching and Feasibility Pump enhanced by cut addition in each pumping cycle. Our solver reduces both the time and the number of nodes for larger instances compared to CPLEX. This effect is far more evident in the $(3, 2)AP_n$ where the number of constraints increases quadratically compared to the $(3, 1)AP_n$. Further experimentation may offer deeper insights on both the performance of such a solver (or a partially modified one) and on our ability to solve exactly larger-scale instances for different values of k and s.

Acknowledgement

We would like to thank Dr. Trivikram Dokka for providing us with the literature instances of the 3-index assignment problem (Grundel and Pardalos 2005).

Table 1: FP variants - $(3,1)AP_n$: Average integrality gap, cycles, time

Instance		OFP1	OFP2	OFP3	ORFP1	ORFP2	ORFP3
axial25	Gap	257.09	551.78	181.08	270.19	363.25	154.92
	Cycles	7	12	3	21	9	5
	Time	0.17	2.02	0.9	0.46 (0.8)	1.56	1.25
axial54	Gap	517.22	729.58	985.86	520.72	1062.69	485.25
	Cycles	11	7	4	8	11	4
	Time	12 (0.6)	65.06	85.31	10.42 (0.8)	95.56	46.26
axial66	Gap	588.91	1233.59	1165.46	814.86	634.44	477.92
	Cycles	5	14	4	7	4	4
	Time	12.3 (0.8)	287.51	251.12	23.86	122.41	122.54
axial80	Gap	647.02	1364.05	1392.16	818.07	1290.83	747.96
	Cycles	5	7	4	8	10	4
	Time	31.68	466.15	581.06	49.38 (0.8)	592.23	310.14
bs25	Gap	216.54	190.73	92.73	69.15	180.61	40.07
	Cycles	50	7	3	3	20	3
	Time	0.96 (0.8)	1.34	0.89	0.1 (0.4)	2.97	0.9
bs54	Gap	61.57	38.52	85.18	28.71	81.11	20.37
	Cycles	4	4	4	4	10	4
	Time	5.33 (0.8)	42.7	63.93	5.7 (0.8)	81.81	48.61
bs66	Gap	63.25	70.3	47.88	35.35	35.15	25.76
	Cycles	11	8	4	6	7	4
	Time	22.96 (0.8)	187.32	185.77	16.83 (0.6)	178.88	133.58
bs80	Gap	53.75	48 .0	28.25	34.06	45.5	19.25
	Cycles	10	5	4	10	9	3
	Time	38.07(0.4)	366.77	498.53	48.88 (0.8)	535.71	260.34
gp25	Gap	0.87	2.36	2.32	0.72	4.26	0.71
	Cycles	1	2	2	1	31	1
	Time	0.1 (0.8)	0.62	0.63	0.1	4.43	0.52
gp54	Gap	0.88	1.85	1.1	0.88	0.67	0.42
	Cycles	3	3	1	2	7	1
	Time	3.42	25.33	20.04	2.97	42.18	14.68
gp66	Gap	0.63	0.63	0.78	0.63	1.34	0.78
	Cycles	1	1	1	2	6	2
	Time	8.55 (0.8)	61.26 (0.8)	89.38	11.36 (0.8)	144.34 (0.8)	68.23
gp80	Gap	1.29	1.2	1.02	0.95	1.22	0.68
	Cycles	29	3	2	14	7.6	3.2
	Time	120.06(0.4)	261.38	302.21	74.96 (0.8)	440.67	236.08

Table 2: Exact algorithms - $(3,1)AP_n$: Average number of nodes, time, integrality gap

Instance		CPLEX	SoS-I	SoSCuts	SoSCutsORFP3
axial25	Nodes	627	565.4	429	520.4
	Time	1.21	1.12	0.98	1.13
	Gap	0	0	0	0
axial54	Nodes	368075	335959	354046	313483
	Time	4608.23	4073.78	5126.36	4283.09
	Gap	0	0	0	0
axial66	Nodes	445512	400366	359688	366384
	Time	$3h$	$3h$	$3h$	$3h$
	Gap	34.24	38.2	29.97	31.79
axial80	Nodes	179122	180121	167720	173515
	Time	$3h$	$3h$	$3h$	$3h$
	Gap	133.07	92.33	106.53	74.02
bs25	Nodes	399	247	205	187
	Time	0.89	0.65	0.67	40.69
	Gap	0	0	0	0
bs54	Nodes	64	99	80	72.8
	Time	4.93	7.99	8.51	27.99
	Gap	0	0	0	0
bs66	Nodes	0	0	0	0
	Time	7.22	12.26	14.77	192.14
	Gap	0	0	0	0
bs80	Nodes	7270	586	465	446
	Time	686.67	72.28	59.16	57.79
	Gap	0	0	0	0
gp25	Nodes	0	0	0	0
	Time	0.39	0.2	0.19	0.67
	Gap	0	0	0	0
gp54	Nodes	0	0	0	0
	Time	9.31	3.92	4.61	18.08
	Gap	0	0	0	0
gp66	Nodes	0	0	0	0
	Time	20.82	14.05	12.41	79.48
	Gap	0	0	0	0
gp80	Nodes	0	0	0	0
	Time	46.72	33.19	30.63	704.52
	Gap	0	0	0	0

Table 3: FP variants - $(3,2)AP_n$: Average integrality gap, cycles, time

Instance		OFP1	OFP2	OFP3	ORFP1	ORFP2	ORFP3
$n = 10$	Gap	8.89	8.51	8.00	8.85	8.21	7.77
	Cycles	3	3	2	5	4	4
	Time	0.02	0.02	0.07	0.02	0.02	0.12
$n = 20$	Gap	27.66	24.99	24.72	29.78	21.8	20.54
	Cycles	6	5	4	18	7	6
	Time	6.21	5.21	14.08	13.07	8.07	22.46
$n = 30$	Gap	42.69	40.61	37.35	32.87	34.11	33.48
	Cycles	9	6	6	10	8	8
	Time	35.18	22.8	241.57	44.25	35.42	374.09
$n = 40$	Gap	55.95	54.86	52.87	48.94	43.65	47.09
	Cycles	11	8	8	13	9	10
	Time	210.92	144.59	1869.09	298.04	211.74	2701.91

Table 4: Exact algorithms - $(3,2)AP_n$: Average number of nodes, time, integrality gap

Instance		CPLEX	SoS-I	SoSCuts	SoSCutsORFP3
$n = 10$	Nodes	813	695	597	624
	Time	2.9	1.66	1.6	1.92
	Gap	0	0	0	0
$n = 20$	Nodes	403464	350702	300106	296380
	Time	$3h$	$3h$	$3h$	$3h$
	Gap	24.47	27.63	29.25	17.07
$n = 30$	Nodes	42738	37188	18029	18030
	Time	$3h$	$3h$	$3h$	$3h$
	Gap	89.36	80.39	87.51	26.20
$n = 40$	Nodes	11986	10266	7927	5927
	Time	$3h$	$3h$	$3h$	$3h$
	Gap	159.68	137.33	160.38	46.92

References

Achterberg, T., and Berthold, T. 2007. Improving the feasibility pump. *Discrete Optimization* 4(1):77–86.

Appa, G.; Magos, D.; and Mourtos, I. 2004. A branch & cut algorithm for a four-index assignment problem. *Journal of the Operational Research Society* 55(3):298–307.

Appa, G.; Magos, D.; and Mourtos, I. 2006. On multi-index assignment polytopes. *Linear Algebra and its Applications* 416(2-3):224–241.

Balas, E., and Qi, L. 1993. Linear-time separation algorithms for the three-index assignment polytope. *Discrete Applied Mathematics* 43(1):1–12.

Balas, E., and Saltzman, M. J. 1989. Facets of the three-index assignment polytope. *Discrete Applied Mathematics* 23(3):201–229.

Balas, E., and Saltzman, M. J. 1991. An algorithm for the three-index assignment problem. *Operations Research* 39(1):150–161.

Bandelt, H.-J.; Crama, Y.; and Spieksma, F. C. 1994. Approximation algorithms for multi-dimensional assignment problems with decomposable costs. *Discrete Applied Mathematics* 49(1):25–50.

Bertacco, L.; Fischetti, M.; and Lodi, A. 2007. A feasibility pump heuristic for general mixed-integer problems. *Discrete Optimization* 4(1):63–76.

Boland, N. L.; Eberhard, A. C.; Engineer, F.; and Tsoukalas, A. 2012. A new approach to the feasibility pump in mixed integer programming. *SIAM Journal on Optimization* 22(3):831–861.

Burkard, R. E.; Rudolf, R.; and Woeginger, G. J. 1996. Three-dimensional axial assignment problems with decomposable cost coefficients. *Discrete Applied Mathematics* 65(1):123–139.

Casselgren, C. J., and Häggkvist, R. 2013. Completing partial latin squares with one filled row, column and symbol. *Discrete Mathematics* 313(9):1011–1017.

Crama, Y., and Spieksma, F. C. 1992. Approximation algorithms for three-dimensional assignment problems with triangle inequalities. *European Journal of Operational Research* 60(3):273 – 279.

De Santis, M.; Lucidi, S.; and Rinaldi, F. 2013. A new class of functions for measuring solution integrality in the feasibility pump approach. *SIAM Journal on Optimization* 23(3):1575–1606.

De Santis, M.; Lucidi, S.; and Rinaldi, F. 2014. Feasibility pump-like heuristics for mixed integer problems. *Discrete Applied Mathematics* 165:152–167.

Dichkovskaya, S., and Kravtsov, M. K. 2006. Investigation of polynomial algorithms for solving the three-index planar assignment problem. *Computational Mathematics and Mathematical Physics* 46(2):212–217.

Dumitrescu, I.; Ropke, S.; Cordeau, J.; and Laporte, G. 2010. The traveling salesman problem with pickup and delivery: polyhedral results and a branch-and-cut algorithm. *Mathematical Programming* 121(2):269–305.

Euler, R.; Burkard, R. E.; and Grommes, R. 1986. On latin squares and the facial structure of related polytopes. *Discrete mathematics* 62(2):155–181.

Euler, R. 2010. On the completability of incomplete latin squares. *European Journal of Combinatorics* 31(2):535–552.

Fischetti, M., and Salvagnin, D. 2009. Feasibility pump 2.0. *Mathematical Programming Computation* 1(2-3):201–222.

Fischetti, M.; Glover, F.; and Lodi, A. 2005. The feasibility pump. *Mathematical Programming* 104(1):91–104.

Frieze, A., and Yadegar, J. 1981. An algorithm for solving 3-dimensional assignment problems with application to scheduling a teaching practice. *Journal of the operational research society* 989–995.

Frieze, A. 1983. Complexity of a 3-dimensional assignment problem. *European Journal of Operational Research* 13(2):161–164.

Gimadi, E. K., and Glazkov, Y. V. 2007. An asymptotically exact algorithm for one modification of planar three-index assignment problem. *Journal of Applied and Industrial Mathematics* 1(4):442–452.

Grundel, D. A., and Pardalos, P. M. 2005. Test problem generator for the multidimensional assignment problem. *Computational Optimization and Applications* 30(2):133–146.

Hansen, P., and Kaufman, L. 1973. A primal-dual algorithm for the three-dimensional assignment problem. *Cahiers du CERO* 15:327–336.

Higgins, M. J. 2013. *Applications of Integer Programming Methods to Solve Statistical Problems*. Ph.D. Dissertation, University of California, Berkeley.

Hooker, J. N. 2012. *Integrated methods for optimization*, volume 100. Springer.

Karapetyan, D., and Gutin, G. 2011. Local search heuristics for the multidimensional assignment problem. *Journal of Heuristics* 17(3):201–249.

Laywine, C., and Mullen, G. 1998. Discrete mathematics using latin squares.

Lysgaard, J.; Letchford, A. N.; and Eglese, R. W. 2004. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming* 100(2):423–445.

Magos, D., and Miliotis, P. 1994. An algorithm for the planar three-index assignment problem. *European Journal of Operational Research* 77(1):141–153.

Magos, D., and Mourtos, I. 2009. Clique facets of the axial and planar assignment polytopes. *Discrete Optimization* 6(4):394–413.

Magos, D., and Mourtos, I. 2013. A characterization of odd-hole inequalities related to latin squares. *Optimization* 62(9):1169–1201.

Magos, D. 1996. Tabu search for the planar three-index assignment problem. *Journal of Global Optimization* 8(1):35–48.

Martello, S., and Toth, P. 1987. Linear assignment problems. *Annals of Discrete Mathematics* 31:259–282.

Padberg, M. W. 1973. On the facial structure of set packing polyhedra. *Mathematical programming* 5(1):199–215.

Pierskalla, W. P. 1968. Letter to the editorthe multidimensional assignment problem. *Operations Research* 16(2):422–431.

Poore, A. B., and Gadaleta, S. 2006. Some assignment problems arising from multiple target tracking. *Mathematical and Computer Modelling* 43(9):1074–1091.

Qi, L.; Ballas, E.; and Gwan, G. 1994. A new facet class and a polyhedral method for the three-index assignment problem. In *Advances in Optimization and Approximation*. Springer. 256–274.

Soicher, L. H. 2013. Optimal and efficient semi-latin squares. *Journal of Statistical Planning and Inference* 143(3):573–582.

Taouil, M., and Hamdioui, S. 2011. Layer redundancy based yield improvement for 3d wafer-to-wafer stacked memories. In *European Test Symposium (ETS), 2011 16th IEEE*, 45–50. IEEE.

Vlach, M. 1967. Branch and bound method for the 3-index assignment problem. *Ekonomicko-Matematicky Obzor* 3(2):181–191.

Walteros, J. L.; Vogiatzis, C.; Pasiliao, E. L.; and Pardalos, P. M. 2014. Integer programming models for the multidimensional assignment problem with star costs. *European Journal of Operational Research* 235(3):553–568.