Financial Portfolio Optimisation

Pierre Flener[1], Justin Pearson[1], and Luis G. Reyna[2]

[1] Department of Information Technology, Uppsala University
Box 337, 751 05 Uppsala, Sweden
Pierre.Flener@it.uu.se, Justin.Pearson@it.uu.se
[2] Global Private Investment Advisory Group, Merrill Lynch
New York, NY 10281-1307, USA
Luis_Reyna@ml.com

Abstract. We give an approximate and often extremely fast method of solving a portfolio optimisation (PO) problem in financial mathematics, which has applications in the credit derivatives market. Its corresponding satisfaction problem is closely related to the balanced incomplete block design (BIBD) problem. However, typical PO instances are an order of magnitude larger than the largest BIBDs solved so far by global search. Our method is based on embedding sub-instances into the original instance. Their determination is itself a CSP. This allows us to solve a typical PO instance, with over 10^{746} symmetries. The high quality of our approximate solutions can be assessed by comparison with a tight lower bound on the cost. Also, our solutions sufficiently improve the currently best ones so as to often make the difference between having or not having a feasible transaction due to investor and rating-agency constraints.

1 Introduction

The structured credit market has seen two new products over the last decade: credit derivatives and credit default obligations (CDOs). These new products have created the ability to leverage and transform credit risk in ways not possible through the traditional bond and loan markets.

CDOs typically consist of a special purpose vehicle that has credit exposure to around one hundred different issuers. Such vehicles purchase bonds and loans and other financial assets through the issuance of notes or obligations with varying levels of risk. In a typical structure, credit losses in the underlying pool are allocated to the most subordinated obligations or notes first. A natural progression of the market has been to use notes from existing CDOs as assets into a new generation of CDOs, called CDO Squared or CDO of CDO [9].

The credit derivatives market has allowed a more efficient mechanism for creating CDO Squared. The idea is to use sub-pools of credit default swaps instead of notes. The sub-pools are chosen from a collection of credits with the level of liquidity and risk adequate to the potential investors. These transactions are sometimes labelled synthetic CDO Squared.

In the creation of a synthetic CDO, the natural question arises on how to maximise the diversification of the sub-pools given a limited universe of previously chosen credits. In a typical CDO Squared, the number of available credits ranges from 250 to 500 and the number of sub-pools from 4 to as many as 25. The investment banker arranging for a CDO Squared usually seeks to maximise the return of the subordinated notes under the constraints imposed by the rating agencies and the investors. This is a challenge that typically is only partially addressed, in part due to the difficulty of pricing the underlying assets [5].[3]

In this paper, we analyse the already financially relevant abstracted problem of selecting the credits comprising each of the sub-pools with a minimal overlap, or maximum diversification. The minimisation of the overlap usually results in better ratings for the notes, typically resulting in more efficient structures.

The remainder of this paper is organised as follows. In Section 2, we discuss the well-known problem of balanced incomplete block design (BIBD), which is related to portfolio optimisation. In Section 3, we formulate the portfolio optimisation (PO) problem, which is an optimisation problem, and show its relationship to the BIBD problem, which is a satisfaction problem. Since the known methods of solving BIBD instances by global search do not scale for the solution of typical instances of the satisfaction version of the PO problem, we introduce in Section 4 a method of approximately solving the PO problem, using a notion of embedding small occurrences of an instance in a larger one. Finally, in Section 5, we conclude, discuss related work, and outline future work.

2 Balanced Incomplete Block Designs

Let V be any set of v elements, called *varieties*. Let $B = \{1, \ldots, b\}$. A *balanced incomplete block design* (BIBD) is a bag of b subsets $B_j \subseteq V$, called *blocks*, each of size k:

$$\forall j \in B : |B_j| = k \tag{1}$$

with $2 \leq k < v$,[4] such that each pair of distinct varieties occurs together in exactly λ blocks. Let V_i be the set of the identifiers of the blocks in which variety i occurs: $V_i = \{j \in B \mid i \in B_j\}$. The V_i are here called *co-blocks*. The previous *balancing condition* can now be formulated by requiring that any two distinct co-blocks intersect over exactly λ elements:

$$\forall\, i_1 \neq i_2 \in V : |V_{i_1} \cap V_{i_2}| = \lambda \tag{2}$$

An implied constraint is that each co-block has the *same* number r of elements, whose value can be determined:

$$\forall i \in V : |V_i| = r = \frac{\lambda \cdot (v-1)}{k-1} \tag{3}$$

[3] There are very few publicly accessible papers we can cite in this introduction, as most are confidential due to the potential financial value of their results.

[4] If $k = v$, then it is a *complete* block design.

This constraint and the already mentioned $2 \leq k < v$ imply that none of the co-blocks can be equal:

$$\lambda < r \tag{4}$$

A further implied constraint is that the co-blocks and blocks have together the same number of elements:

$$v \cdot r = b \cdot k \tag{5}$$

These implied constraints are insufficient *existence conditions* for a BIBD.

A BIBD is thus parameterised by a 5-tuple $\langle v, b, r, k, \lambda \rangle$ of parameters, any three of which are independent. Originally intended for the design of statistical experiments, BIBDs also have applications in cryptography and other domains. See [1], or `http://mathworld.wolfram.com/BlockDesign.html`, or Problem 28 at `http://www.csplib.org/` for more information.

Blocks and co-blocks are dual: an alternative formulation is that a BIBD is a *set* of v subsets $V_i \subseteq B$, each of size r, such that the preceding constraints (1) to (5) hold, where block B_j is then the set of varieties comprising it, that is $B_j = \{i \in V \mid j \in V_i\}$.

One way of modelling a BIBD is in terms of its *incidence matrix*, which is a $v \times b$ matrix, such that the entry at the intersection of row i and column j is 1 if $i \in B_j$ (that is $j \in V_i$) and 0 otherwise. The first three constraints are then modelled by requiring, respectively, that there are exactly k ones (that is a sum of k) for each column, a scalar product of exactly λ for any pair of distinct rows, and exactly r ones (that is a sum of r) for each row.

Since the varieties and blocks are indistinguishable, any two rows or columns of the incidence matrix can be freely permuted. Breaking all the resulting $v! \cdot b!$ symmetries can in theory be performed, for instance by $v! \cdot b! - 1$ (anti-) lexicographical ordering constraints between vectors extracted from the incidence matrix [4, 8]. In practice, strictly anti-lexicographically ordering (denoted by $>_{lex}$) the rows (since co-blocks cannot be repeated) as well as anti-lexicographically ordering (denoted by $\geq_{lex}$) the columns (since blocks can be repeated) works quite fine, due to the balancing constraint (2) [7], especially when labelling in a row-wise fashion and trying the value 1 before the value 0. This much improves the best previously reported results under global search and allows the solution of previously unsolved instances. By simultaneously performing symmetry-breaking during search in the SBDD style [6], but augmenting it with group-theoretical insights and some heuristics, improvements of another order of magnitude can be achieved, but only when computing all the solutions [12]. The instances solved in [12] with $4 \leq v \leq 25$, which is the range of interest to us, have values of b up to 50, which is an order of magnitude below our range of interest.

3 Portfolio Optimisation

After precisely formulating the portfolio optimisation (PO) problem of the introduction and exhibiting its relationship to the BIBD problem, we derive an important implied constraint for the PO problem, before showing how to model it and how to exactly solve sub-real-life-scale instances thereof.

3.1 Formulation

The *portfolio optimisation* (PO) problem is formulated as follows. Let $V = \{1, \ldots, v\}$ and let $B = \{1, \ldots, b\}$ be a set of credits. A *portfolio* is a set of v subsets $V_i \subseteq B$, called *sub-pools*, each of size r:

$$\forall i \in V : |V_i| = r \tag{6}$$

such that the maximum intersection size of any two distinct sub-pools is minimised. A portfolio is thus parameterised by a 3-tuple $\langle v, b, r \rangle$ of independent parameters. By abuse of language, $\langle v, b, r \rangle$ denotes even sub-optimal solutions.

There is a universe of about $250 \leq b \leq 500$ credits. A typical portfolio contains about $4 \leq v \leq 25$ sub-pools, each of size $r \approx 100$.

Note that we have formulated the PO problem using the same notation as for the BIBD problem. The relationship with the (co-block formulation of the) BIBD problem is indeed striking, with credits taking the role of the block identifiers, sub-pools taking the role of the co-blocks, and the co-block size being fixed, as per the related constraints (3) and (6). But the similarity ends there, as the BIBD balancing condition (2) refers to a constant λ as the co-block intersection size, while the maximum co-block intersection size is to be minimised in a portfolio. In other words, the BIBD problem is a constraint satisfaction problem (CSP), while the PO problem is a constraint optimisation problem (COP). Also, the typical value of b for a portfolio is an order of magnitude larger than what has been tried so far with global search for BIBDs [12].

For syntactic continuity, let us call λ the maximum of the intersection sizes in a portfolio. This gives us the following PO constraint, related to the BIBD constraint (2):

$$\forall\ i_1 \neq i_2 \in V : |V_{i_1} \cap V_{i_2}| \leq \lambda \tag{7}$$

where λ is then the cost expression that is to be minimised:

$$\text{minimise } \lambda \tag{8}$$

with $\lambda \leq r$ (note the difference with the BIBD implied constraint (4)).

We parameterise a PO CSP by a 4-tuple $\langle v, b, r, \lambda \rangle$ of independent parameters, where λ need not be the minimal value. Note that PO CSPs with $\lambda = r$ are trivial to construct, as it suffices to make all co-blocks equal.

3.2 An Implied Constraint

We now show how to derive a tight lower bound on λ for the PO problem, and argue why the PO problem does not (seem to) have a counterpart of the BIBD constraint (1) on the block sizes, and hence not a counterpart of the BIBD implied constraint (5). The following theorem exactly fits the requirements of the PO problem, provided *all* the credits are used in the portfolio, which is often a realistic assumption:

Theorem 1 (Corrádi [2, 10]). *Let $V_1, \ldots, V_v$ be r-element sets and B be their union. If $|V_{i_1} \cap V_{i_2}| \leq \lambda$ for all $i_1 \neq i_2$, then*

$$|B| \geq \frac{r^2 \cdot v}{r + (v-1) \cdot \lambda}$$

Since $|B| = b$ here, we get as a PO implied constraint a tight lower bound on λ by rearranging the previous formula and rounding up so that λ is a natural number:[5]

$$\lambda \geq \left\lceil \frac{r \cdot (r \cdot v - b)}{b \cdot (v-1)} \right\rceil \quad \wedge \quad \lambda \geq 0 \tag{9}$$

The lower bound predicted by this constraint is not always exact, as shown in the following example.

Example 1. For $\langle 10, 8, 3 \rangle$, we obtain $\lambda \geq \lceil \frac{11}{12} \rceil$, hence $\lambda \geq 1$. For $\langle 9, 8, 3 \rangle$, we obtain $\lambda \geq \lceil \frac{57}{64} \rceil$, hence $\lambda \geq 1$. However, it is not difficult to show (with the method to be shown in Section 3.3) that there are no 10 or even 9 subsets of size 3 in an 8-element set such that they intersect pairwisely over at most $\lambda = 1$ element. In fact, these two instances are at best solved with $\lambda = 2$; some of the sets of such optimal solutions pairwisely intersect over only 1 element. (This example will be continued in Example 2.)

It is tempting to think that tight bounds can be similarly obtained on the block sizes. Indeed, a portfolio $\langle v, b, r \rangle$ becomes a BIBD if b divides $v \cdot r$ and if all the sub-pools must have pairwise intersections of *exactly* (rather than at most) λ elements: the integer value $k = \frac{v \cdot r}{b}$ is then obtained via the BIBD implied constraint (5). In case b does not divide $v \cdot r$, one may be tempted to adjust the portfolio parameters first. However, BIBDs of the size considered here, namely for $250 \leq b \leq 500$ blocks, are about one order of magnitude larger than what has been tried so far in global search, and our experiments suggest that those methods do not scale to BIBDs of that size, especially that the BIBD existence conditions are very weak. Also, no PO constraint forces the credits to spread in some manner over the sub-pools, so that neither $\lceil \frac{v \cdot r}{b} \rceil$ is an upper bound on k, nor $\lfloor \frac{v \cdot r}{b} \rfloor$ is a lower bound on k. Indeed, we have designed portfolios where the block sizes are distributed over the entire $1, \ldots, v$ range (see Example 2).

It is also tempting to think that it is sufficient (and easier) to find sub-pools whose pairwise intersections are of size exactly λ, rather than upper bounded by λ. However, there is no solution to $\langle 10, 8, 3 \rangle$ where the pairwise intersection sizes are all equal to $\lambda = 2$, whereas Example 1 establishes the existence of a solution where the pairwise intersection sizes are upper bounded by $\lambda = 2$.

3.3 Modelling and Exact Solution

One way of modelling a portfolio is in terms of its *incidence matrix*, which is a $v \times b$ matrix, such that the entry at the intersection of row i and column j is

[5] The same bound can be obtained by injecting the resolution of the BIBD implied constraint (5) for k into the BIBD implied constraint (3) and then resolving for λ.

	blocks/credits							
1	1	1	1	0	0	0	0	0
2	1	1	0	1	0	0	0	0
3	1	1	0	0	1	0	0	0
4	1	1	0	0	0	1	0	0
5	1	1	0	0	0	0	1	0
6	1	1	0	0	0	0	0	1
7	1	0	1	1	0	0	0	0
8	1	0	1	0	1	0	0	0
9	1	0	1	0	0	1	0	0
10	1	0	1	0	0	0	1	0

Table 1. An optimal solution to $\langle 10, 8, 3\rangle$, with cost $\lambda = 2$. The rows correspond to the co-blocks (sub-pools).

1 if $j \in V_i$ and 0 otherwise. The PO constraints (6) and (7) are then modelled by requiring, respectively, that there are exactly r ones (that is a sum of r) for each row and a scalar product of at most λ for any pair of distinct rows.

The following example gives an optimal portfolio under this model, and uses it to show that the PO problem does not enjoy the optimal sub-structure property.

Example 2. (Continuation of Example 1.) An optimal solution to $\langle 10, 8, 3\rangle$, with cost $\lambda = 2$, is given in Table 1. Note that the block sizes are distributed over the entire $1, \ldots, v$ range, namely one block each of sizes 1, 5, 6, 10, and four blocks of size 2. Now, for $\langle 8, 8, 3\rangle$, we obtain $\lambda \geq \lceil \frac{6}{7} \rceil$, hence $\lambda \geq 1$, and it turns out that there *are* 8 subsets of size 3 in an 8-element set such that they intersect pairwisely over at most 1 element. We can now see why the PO problem does not enjoy the optimal sub-structure property, namely that an optimal solution to an instance does not necessarily contain optimal solutions to sub-instances. Indeed, the optimal solution to $\langle 10, 8, 3\rangle$ in Table 1, with cost 2, contains no 8 subsets of size 3 in the 8-element set such that they intersect pairwisely over at most 1 element. Note that the last 4 sets each have pairwise intersections of size 1 with 4 of the first 6 sets, while all other pairwise intersections are of size 2.

The tight lower bound on the cost expression λ suggests a (naive) method of exactly solving (small instances of) the PO COP as a sequence of PO CSPs: set λ to some value "comfortably" above that tight lower bound, and lower it by 1 each time that CSP has a solution.

The sub-pools are indistinguishable, and we assume (in a first approximation) that all the credits are indistinguishable. Hence any two rows or columns of the incidence matrix can be freely permuted. Breaking all the resulting $v! \cdot b!$ symmetries can in theory be performed, for instance by $v! \cdot b! - 1$ (anti-) lexicographical ordering constraints [4]. In practice, in the CSP version of the PO problem (where a value for λ is given), strictly anti-lexicographically ordering the rows (since sub-pools cannot be repeated in portfolios with $\lambda < r$) as well as anti-lexicographically ordering the columns (since credits can appear in the

same sub-pools) works quite fine for values of b up to about 36, due to the constraint (7), especially when labelling in a row-wise fashion and trying the value 1 before the value 0. However, this is one order of magnitude below the typical value for b in a portfolio. Also, the absence of a constraint on the block sizes makes $\langle v, b, r, \lambda \rangle$ much harder to solve than $\langle v, b, r, k, \lambda \rangle$, if such a k exists. Hence another method than this BIBD-style approach is necessary, or we need to design approximately optimal portfolios, as discussed next.

4 Approximate Solution to Portfolio Optimisation

Our method of efficiently finding possibly approximate solutions to the portfolio optimisation (PO) problem rests on two key insights, explained first.

4.1 Underconstrainedness

The first insight comes from observing that the typical values of v (the number of sub-pools) are quite small for the typical values of b (the number of credits) and r (the size of the sub-pools), as shown in the following example.

Example 3. The first three columns of Table 2 chart how the lower bound on λ evolves with $v \geq 2$ according to the PO implied constraint (9) when $b = 350$ and $r = 100$. The lower bound on λ initially grows from 0 for $v = 2$, to between 5 and 26 for the typical values of v (which are between 4 and 25), but does not grow much after that; in fact, it never exceeds 29, which it reaches for $v = 127$. This effect is exacerbated for smaller values of b and r, as shown in the fourth and fifth columns of Table 2.

While this example illustrates a prediction weakness of Theorem 1 for large values of v, the main lesson is that there is a range for v in which the lower bound on λ does not change quickly for fixed values of b and r. For the ranges of values of v, b, and r that are of interest here, v is within that zone.

The consequence is that the PO problem instances of interest here seem underconstrained in the sense that one may get (many) more than the intended v sub-pools of the same size r from the same universe of b credits, without seeing the maximum intersection size of the sub-pools increase. Dually, one may draw the intended v sub-pools of the same size r from a (much) smaller universe than the available b credits, without seeing the maximum intersection size of the sub-pools increase. For instance, Theorem 1 predicts that $v = 10$ sub-pools of $r = 100$ credits each may be drawn with a maximum intersection size of 21 from a universe of $347 \leq b \leq 357$ credits. Again, this effect is exacerbated for smaller values of b and r. This underconstrainedness may lead to considerable combinatorial explosion. In fact, we have been unable to solve any PO CSP instances of the magnitude considered here with the BIBD-style method outlined in Section 3.3, even when setting a quite high value for λ and allocating an entire CPU week. Labelling just one row of the incidence matrix already tends to take a lot of time after the first few rows.

	$b = 350$ and $r = 100$		$b = 35$ and $r = 10$			
v	unrounded lower bound on λ	rounded lower bound on λ	unrounded lower bound on λ	rounded lower bound on λ	time to first solution	backtracks to first solution
2	-42.86	0	-4.286	0	0.01	0
3	-7.14	0	-0.714	0	0.04	0
4	4.76	5	0.476	1	0.09	1
5	10.71	11	1.071	2	0.26	184
6	14.28	15	1.428	2	0.74	658
7	16.67	17	1.667	2	1.23	921
8	18.37	19	1.837	2	4.89	8872
9	19.64	20	1.964	2	? + 0.85	? + 566
10	20.63	21	2.063	3	1.40	567
11	21.43	22	2.143	3	1.62	567
12	22.08	23	2.208	3	2.07	663
13	22.62	23	2.262	3	3.01	1878
14	23.08	24	2.308	3	3.80	2038
15	23.47	24	2.347	3	4.82	2245
16	23.81	24	2.381	3	9.94	9331
17	24.11	25	2.411	3	12.97	10221
...		25		3		
22	25.17	26	2.517	3	39.59	16078
...		26		3		
29	26.02	27	2.602	3	117.72	35305
...		27		3		
47	27.02	28	2.702	3	?	?
...		28		3		
127	28.01	29	2.801	3	?	?
...		29		3		

Table 2. Unrounded and rounded lower bounds on the maximum intersection size λ for $v \geq 2$ co-blocks and b blocks of size r, as given by the PO implied constraint (9).

4.2 Embeddings

The second insight is that computing optimal solutions is not always practical. As shown below, we can often very efficiently solve real-life PO problem instances with values for λ that are within 5% of, if not identical to, the lower bound given by the PO implied constraint (9). Since that lower bound is not always exact, and since there is currently no better or faster way of solving real-life PO problem instances, our results are sufficient. Some may even turn out to be optimal. So we investigate the approximate solution of real-life PO problem instances. The idea is to embed small PO problem instances within a large, real-life one, as illustrated in the following example.

Example 4. We can embed 10 occurrences of $\langle 10, 35, 10\rangle$ within $\langle 10, 350, 100\rangle$. A not necessarily optimal solution to the PO COP $\langle 10, 350, 100\rangle$ can be built by making 10 copies of each column in any possibly optimal solution to the PO

COP $\langle 10, 35, 10\rangle$. The fifth column of Table 2 gives $\lambda \geq 3$ for the PO COP $\langle 10, 35, 10\rangle$. Solving the PO CSP $\langle 10, 35, 10, 3\rangle$ with the BIBD-style method outlined in Section 3.3 is a matter of about one CPU second and 567 backtracks, and such a portfolio does exist. Since $10 \cdot 3 = 30$, this means that we can build from it a solution to the PO CSP $\langle 10, 350, 100, 30\rangle$. Since the third column of Table 2 gives $\lambda \geq 21$ for the PO COP $\langle 10, 350, 100\rangle$, the built solution with cost $\lambda = 30$ is quite far above that lower bound and may thus be sub-optimal. (This example will be continued in Example 6.)

This kind of embedding is a standard concept for BIBDs. Indeed, a BIBD $\langle v, b, r, k, \lambda\rangle$ is said to be an *m-multiple BIBD* if $\langle v, \frac{b}{m}, \frac{r}{m}, k, \frac{\lambda}{m}\rangle$ parameterises a BIBD under the constraints (1) to (5) [1]. In other words, shrinking the number of blocks by a factor m shrinks the sizes of the co-blocks and their intersections by the same factor m (provided they all divide m). Since there are no existence conditions for portfolios, whose design is a COP rather than a CSP, the corresponding concept for portfolios has an easier definition, given next.

Definition 1. *A portfolio $\langle v, b, r\rangle$ is an* m-multiple portfolio *if m divides both b and r. We denote this by $\langle v, b, r\rangle = m \cdot \langle v, \frac{b}{m}, \frac{r}{m}\rangle$.*

For the same reason, we can only compare the predicted lower bounds on the maximum sub-pool intersection sizes, rather than the actual intersection sizes as for BIBDs. The following property establishes that the same ratio holds between those lower bounds for portfolios and their multiples.

Property 1. The PO implied constraint (9) predicts $\lambda \geq \lceil \mu \rceil$ for $\langle v, b, r\rangle$ if and only if it predicts $\lambda \geq \lceil \frac{\mu}{m} \rceil$ for $\langle v, \frac{b}{m}, \frac{r}{m}\rangle$.

Example 5. We have $\langle 10, 350, 100\rangle = 10 \cdot \langle 10, 35, 10\rangle$. Table 2 confirms the ratio of 10 between the unrounded lower bounds on λ for the two involved instances.

However, a portfolio is not always an exact multiple of another portfolio. Rather than adjusting the size of a desired portfolio so that it becomes a multiple of another portfolio, we advocate generalising the notion of multiples of a design and here do so for portfolios. Let us first show the intuition on an example.

Example 6. (Continuation of Example 4.) Reconsider the $\langle 10, 350, 100\rangle$ portfolio. It is not a 12-multiple of any portfolio as 12 does not divide both 350 and 100. Since $350 = 12 \cdot 27 + 26$ and $100 = 12 \cdot 8 + 4$, a not necessarily optimal solution to the PO COP $\langle 10, 350, 100\rangle$ can be built by making 12 copies of each column in any possibly optimal solution to the PO COP $\langle 10, 27, 8\rangle$ and appending any possibly optimal solution to the PO COP $\langle 10, 26, 4\rangle$. The PO implied constraint (9) gives $\lambda \geq 2$ for the PO COP $\langle 10, 27, 8\rangle$ and $\lambda \geq 1$ for the PO COP $\langle 10, 26, 4\rangle$. Solving the PO CSPs $\langle 10, 27, 8, 2\rangle$ and $\langle 10, 26, 4, 1\rangle$ with the BIBD-style method outlined in Section 3.3 is a matter of about 1 CPU second and 69 backtracks total, and such portfolios do exist. Since $12 \cdot 2 + 1 = 25$, this means that we can build from them a solution to the PO CSP $\langle 10, 350, 100, 25\rangle$. Since the third column of Table 2 gives $\lambda \geq 21$ for the PO COP $\langle 10, 350, 100\rangle$, the built solution with cost $\lambda = 25$ is still a bit above that lower bound and may thus be sub-optimal. (This example will be continued in Example 7.)

Let us now formalise all the intuitions from this example.

Definition 2. *A portfolio* $\langle v, b, r\rangle$ embeds m *occurrences of a portfolio* $\langle v, b_1, r_1\rangle$ *and* 1 *occurrence of a portfolio* $\langle v, b_2, r_2\rangle$*, which is denoted by* $\langle v, b, r\rangle = m \cdot \langle v, b_1, r_1\rangle + \langle v, b_2, r_2\rangle$*, if the following three constraints hold:*

$$b = m \cdot b_1 + b_2 \tag{10}$$

$$r = m \cdot r_1 + r_2 \tag{11}$$

$$0 \leq r_i \leq b_i \geq 1 \quad \text{for } i = 1, 2 \tag{12}$$

The constraints (10) and (11) ensure that the embedding is exact. The constraint (12) ensures that the sub-pools can be subsets of the set of credits, for each of the two embedded portfolios. It also eliminates the two cases ($b_i = 0$) where the PO implied constraint (9) cannot be evaluated.

Note that this embedding by vertical division of the incidence matrix is possible because of the full column symmetry of the latter and because no PO constraint works against it. However, an embedding by horizontal division of the incidence matrix will lead to identical rows, that is worst-case solutions ($\lambda = r$).

An upper bound on the cost of an embedding portfolio can be computed from the costs of its embedded portfolios, as shown next.

Property 2. The cost λ of a portfolio embedding m occurrences of a portfolio $\langle v, b_1, r_1\rangle$ of cost λ_1 and one occurrence of a portfolio $\langle v, b_2, r_2\rangle$ of cost λ_2 satisfies the inequality $\lambda \leq m \cdot \lambda_1 + \lambda_2$.

The reason why there may be a strict inequality is that the cost of a portfolio is the maximum of its sub-pool intersection sizes. Consider $v = 3$ and $m = 1$: the first embedded portfolio may have $1, 1, 2$ as intersection sizes, and the second embedded portfolio may have $1, 2, 1$ as intersection sizes, both with a maximum of 2, giving $1 + 1, 1 + 2, 2 + 1$ as intersection sizes for the embedding portfolio, with a maximum of $3 < 1 \cdot 2 + 2 = 4$. For this reason, the calculated cost 25 of the embedding portfolio in Example 6 is in fact an upper bound, rather than the exact cost as stated there. Hence it is in general better to observe the actual cost of the embedding portfolio than to use the upper bound given by Property 2. In this case, observation establishes that the cost is 25.

4.3 Approximate Solution

The issue now becomes how to construct suitable portfolio embeddings, so that near-optimal, if not optimal, real-life-scale portfolios can be designed. We advocate solving the CSP versions of the two embedded instances, setting as λ the rounded lower bound given by the PO implied constraint (9).

Our method takes as additional input a cost Λ that we are trying to undercut, say because it is the cost of the currently best portfolio (or the upper bound on that cost, as determined by Property 2).

Two heuristic constraints on $m, v, b, r, b_1, r_1, b_2, r_2$ in addition to the three constraints of Definition 2 become necessary in order to make the method pragmatic. Let λ_i be the rounded lower bounds given for the two embedded portfolios $\langle v, b_i, r_i \rangle$ by the PO implied constraint (9). The additional constraints are justified and given in the following.

First, we must restrict the focus to the pairs of embedded portfolios that have a chance of leading to a portfolio whose combined cost is lower than Λ:

$$m \cdot \lambda_1 + \lambda_2 < \Lambda \tag{13}$$

Indeed, the left-hand side is by Property 2 the upper bound on the cost of the embedding portfolio built from solutions, if they exist, to the two $\langle v, b_i, r_i, \lambda_i \rangle$ PO CSPs. In practice, it is usually equal to the cost of such an embedding portfolio, hence this constraint. Note that this constraint implies that $m < \Lambda$.

Second, knowing that PO CSPs with values of b up to about 36 can often be solved (quite quickly) using the BIBD-style method outlined in Section 3.3, the objective in choosing the parameters of the embedding is to have *both* embedded instances within that range for b:

$$b_i \leq 36 \quad \text{for } i = 1, 2 \tag{14}$$

Note that the determination of candidate embeddings, which are pairs of CSPs, is thus itself a CSP.

There is no guarantee that all PO CSPs with $b \leq 36$ can be solved sufficiently quickly. For instance, the sixth and seventh columns of Table 2 chart the CPU times in seconds and backtracks for $\langle v, 35, 10, \lambda \rangle$ for $v \geq 2$ and λ equal to the rounded lower bound in the fifth column. The experiments were conducted on a Sun SPARC Ultra Station 10 in our SICStus Prolog 3.10.1 implementation of the BIBD-style method outlined in Section 3.3. A question mark means that we stopped the solution process after a CPU hour. The entry in the row $v = 9$ means that $\langle 9, 35, 10, 2 \rangle$ timed out (in fact, it takes about 25 CPU hours and about $537 \cdot 10^6$ backtracks to fail[6]), while $\langle 9, 35, 10, 3 \rangle$ takes only 0.85 CPU seconds and 566 backtracks to succeed. We observe that for the range of values of v where the rounded lower bound on λ remains the same, the runtimes increase with v. In other words, they increase when the rounding distance for the lower bound on λ decreases. This may not always be the case. The same pattern can be observed for the number of backtracks. The rounding distance seems to be a good indicator of the constrainedness of a PO CSP. A good heuristic then seems to be that we should favour embeddings where both embedded instances have not too small rounding distances. In our observation, for the typical values of v, instances with rounding distances below 0.15 are often problematic. Hence we also advocate ordering the embedded instance pairs that satisfy the

[6] Amazingly, these figures were obtained on the same hardware in our OPL implementation under OPL Studio 3.0.2, which performs *no* symmetry breaking for lack of a lexicographical ordering constraint! We aborted our SICStus Prolog 3.10.1 implementation after several CPU days, both with and without symmetry breaking.

m	$\langle v, b_1, r_1, \lambda_1 \rangle$	unrounded λ_1	$\langle v, b_2, r_2, \lambda_2 \rangle$	unrounded λ_2	$m \cdot \lambda_1 + \lambda_2$
10	$\langle 10, 32, 09, 2 \rangle$	1.812	$\langle 10, 30, 10, 3 \rangle$	2.592	23
11	$\langle 10, 31, 09, 2 \rangle$	1.903	$\langle 10, 09, 01, 1 \rangle$	0.012	23
9	$\langle 10, 36, 10, 2 \rangle$	1.975	$\langle 10, 26, 10, 4 \rangle$	3.162	22
18	$\langle 10, 18, 05, 1 \rangle$	0.988	$\langle 10, 26, 10, 4 \rangle$	3.162	22
19	$\langle 10, 18, 05, 1 \rangle$	0.988	$\langle 10, 08, 05, 3 \rangle$	2.917	22
11	$\langle 10, 30, 09, 2 \rangle$	2.000	$\langle 10, 20, 01, 0 \rangle$	-0.056	22

Table 3. Embeddings of $\langle 10, 350, 100 \rangle$ satisfying the constraints (10) to (14) for $\Lambda = 25$, ordered by decreasing rounding distance for λ_1.

constraints (10) to (14) by decreasing rounding distance for λ_1, so that the apparently easier pairs are attempted first. Setting a time-limit on each attempt is another useful refinement.

Let us now illustrate this method.

Example 7. (Continuation of Example 6.) Let us try and improve the portfolio with cost $\Lambda = 25$ previously obtained for $\langle 10, 350, 100 \rangle = 12 \cdot \langle 10, 27, 8 \rangle + \langle 10, 26, 4 \rangle$. The embeddings satisfying the constraints (10) to (14) are given in Table 3, ordered by decreasing rounding distance for λ_1. Note that none of these embeddings has a combined cost of $\lambda = 21$, which is the lower bound given by the PO implied constraint (9) for $\langle 10, 350, 100 \rangle$. This may be an artifact of the way we define embeddings or of the way we heuristically constrain the embeddings. Setting a time limit of one CPU hour, we attempt to solve the PO CSPs in the second and fourth columns, proceeding row by row.

The first embedding only takes about 13 CPU seconds and 13,152 backtracks total to solve its two PO CSPs. Hence we can build a solution to $\langle 10, 350, 100 \rangle$ from 10 copies of the optimal solution (with $\lambda = 2$) to $\langle 10, 32, 9 \rangle$ and one copy of the optimal solution (with $\lambda = 3$) to $\langle 10, 30, 10 \rangle$; it has an observed cost of exactly $\lambda = 10 \cdot 2 + 3 = 23 > 21$.

The second embedding takes about 47 CPU minutes and about $4 \cdot 10^6$ backtracks (mostly because of the first embedded instance, as the second one has $\lambda_2 = r_2$ and is thus trivial to solve). We get another solution of (predicted and observed) cost $23 = 11 \cdot 2 + 1$.

The third embedding has a first embedded PO CSP that times out, hence we ignore it and move on.

The fourth and fifth embeddings both contain $\langle 10, 18, 5, 1 \rangle$, which fails in about 6 CPU minutes and 345,595 backtracks. Hence $\lambda_1 \geq 2$, and $m \cdot \lambda_1 + \lambda_2$ is at least 40 for the fourth embedding and at least 41 for the fifth embedding, which are both much worse costs than in the currently best solution.

The sixth embedding is very interesting. Its first embedded PO CSP can be solved as a BIBD with blocks of fixed size $k = 3$, as the unrounded λ_1 is a natural number and as b_1 divides $v \cdot r_1$. This additional constraint (1) on the block sizes gives very good propagation, and the BIBD method outlined at the end of Section 2 can solve this instance in about 0.39 CPU seconds and 23 backtracks, whereas the BIBD-style method outlined in Section 3.3 timed out

11 copies of each column of	1 copy of each column of
11111111100000000000000000000	10000000000000000000
11000000011111110000000000000	01000000000000000000
11000000000000011111110000000	00100000000000000000
00110000011000001100000 1110000	00010000000000000000
00110000000110000011000000 1110	00001000000000000000
00001100011000000000 1100001101	00000100000000000000
00001100000001100010001 1100010	00000010000000000000
00000011000110000000 1011010001	00000001000000000000
00000010100001011100000000 1011	00000000100000000000
00000001100000110001010011 0100	00000000010000000000

Table 4. Our currently best solution to $\langle 10, 350, 100\rangle$, built from $11 \cdot \langle 10, 30, 9\rangle + \langle 10, 20, 1\rangle$, and of cost $11 \cdot 2 + 0 = 22 > 21$.

on the corresponding PO CSP, which does not have that constraint. The second embedded PO CSP is trivial (in the sense that there at least as many credits as in the union of the requested sub-pools) since $v \cdot r_2 \leq b_2$ and is solved in about 0.21 CPU seconds and 0 backtracks.

Hence we can build a solution, given in Table 4, to $\langle 10, 350, 100\rangle$ from 11 copies of the optimal solution (with $\lambda = 2$) to $\langle 10, 30, 9\rangle$ and one copy of the optimal solution (with $\lambda = 0$) to $\langle 10, 20, 1\rangle$; it has an observed cost of exactly $\lambda = 11 \cdot 2 + 0 = 22 > 21$. Note that the last 10 credits are not used in this solution. This solution may actually turn out to be optimal, considering the prediction weakness of Theorem 1.

5 Conclusions

Summary. We have given an approximate and often extremely fast method of solving a new portfolio optimisation (PO) problem in financial mathematics. Its corresponding satisfaction problem is closely related to the balanced incomplete block design (BIBD) problem. However, typical PO instances are an order of magnitude larger than the largest BIBDs solved so far by global search, and the PO problem lacks a counterpart of a crucial BIBD constraint. Hence current BIBD-style solving methods are not suitable for real-life PO instances. Our method is based on embedding (multiple copies of) independent sub-instances into the original instance. Their determination is itself a constraint satisfaction problem. The high quality of our approximate solutions can be assessed by comparison with a tight lower bound on the cost.

Generalisation. The generalisation of the main idea is as follows, in the context of a large constraint optimisation problem where a (tight) lower bound on its cost can be somehow calculated. The idea is to embed several independent small problem instances P_i within the given large problem instance P. A feasible solution S to P can then be built from possibly optimal feasible solutions S_i to the P_i. The quality of S can be assessed against the lower bound on the cost of

the optimal solution to P. If there is a relationship between the costs of S and the S_i, then this relationship can be used to determine the P_i via a CSP, using the lower bounds on their costs. For PO, this relationship is given by Property 2.

Related Work. The idea of exploiting *independent* sub-problems also underlies Tree-Based Russian Doll Search [11]. The idea of embedding (multiple copies of) sub-problem instances into a larger problem instance is related to the concept of abstract local search [3], where a concrete solution is built from a solution to an abstraction of the original problem instance and then analysed for flaws so as to infer a new abstract solution. This works well if the concretisation and analysis steps are tractable and if the abstraction is optimality preserving, in the sense that optimal concrete solutions can be built from abstract solutions. Our embedded problem instances can indeed be jointly seen as an *abstraction* of the original problem instance. For instance, entire bundles of credits are here abstracted into single super-credits. We have been unable so far to prove *optimality preservation* of such portfolio abstractions, or to find conditions for it. As also observed in [3], this is not problematic for hard problem instances, such as the typical PO problem instances considered here, where the utility of abstractions can only be assessed by comparison with other techniques. In any case, we have seen that our portfolio abstractions lead to solutions that are extremely close to a tight lower bound on the cost.

Also, we have found only one paper taking a constraint programming approach to portfolio selection [13], but the tackled problem there is actually different from ours and is limited to portfolios consisting of just one sub-pool.

Future Work. Our notion of embeddings can be generalised to any linear combination of several sub-instances. Indeed, Definition 2 is restricted to embeddings of always *two* sub-instances, with coefficients m and 1, respectively. The price to pay for this restriction may have been that a solution of cost 21 eluded us in Example 7, though it may also be that no such solution exists and that our solution of cost 22 is in fact optimal.

Some additional abstraction may reduce the $1, \ldots, v$ range of observed block sizes. Indeed, a counterpart of the BIBD constraint (1) might enormously speed up the solution process. The facts that some PO instances (such as $\langle 9, 35, 10, 2\rangle$) take CPU days to fail while increasing their λ (to obtain $\langle 9, 35, 10, 3\rangle$ here) then leads to quasi instantaneous success, and that other PO instances (such as $\langle 10, 30, 9, 2\rangle$) take CPU hours to succeed while the corresponding BIBD instances, if any ($\langle 10, 30, 9, 3, 2\rangle$ here), quasi instantaneously succeed, show that there is still much space for improving our method. Such an additional abstraction might only come at the price of losing optimal solutions, though.

The run-times and backtrack counts of our implementation of the BIBD-style method outlined in Section 4 can be improved with the additional symmetry-breaking techniques of STAB [12].

Finally, it would be interesting to compare our results with those obtained by other complete-search techniques as well as by local-search techniques.

Conclusion and Financial Relevance. Our optimisation has eliminated the need for ad hoc manual permutations. On average, we have found that the over-

lap in a given portfolio can be decreased anywhere from 2% to 4% by using the current formulation. Even though this may not sound like a dramatic improvement, the ability to reduce the maximum overlap from 25% to 22%, say, may make the difference between having or not having a feasible transaction due to investor and rating-agency constraints.

It should be pointed out that it is easy to reduce the overlap by increasing the number of available credits. However, such new credits tend to be less known and thus more difficult to analyse, resulting in less than efficient portfolios.

In practice, the credits are not all indistinguishable. A client might have personal preferences for or against some credits, declare some credits as mutually exclusive, and so on. The advantage of our deployment of constraint technology is that such specific needs can be neatly added without having to devise new portfolio optimisation algorithms from scratch each time. However, such side constraints may break some of the full column symmetry, so partial symmetry breaking has to be deployed instead. Work in this direction has begun.

Challenge. We challenge the reader to answer the open question whether a $\langle 10, 350, 100 \rangle$ portfolio with optimal cost 21 exists or not.

Acknowledgements. We thank the referees for their useful comments.

References

1. C. J. Colbourn and J. H. Dinitz, editors. *The CRC Handbook of Combinatorial Designs*. CRC Press, 1996.
2. K. Corrádi. Problem at Schweitzer competition. *Mat. Lapok*, 20:159–162, 1969.
3. J. M. Crawford, M. Dalal, and J. P. Walser. Abstract local search. In *Proceedings of the AIPS'98 Workshop on Planning as Combinatorial Search*, 1998.
4. J. M. Crawford, M. Ginsberg, E. Luks, and A. Roy. Symmetry-breaking predicates for search problems. In *Proceedings of KR'96*, pages 148–159, 1996.
5. S. Das and G. Geng. Correlated default processes: A criterion-based copula approach. *Journal of Investment Management*, forthcoming.
6. T. Fahle, S. Schamberger, and M. Sellmann. Symmetry breaking. In T. Walsh, editor, *Proc. of CP'01*, volume 2293 of *LNCS*, pages 93–107. Springer-Verlag, 2001.
7. P. Flener, A. M. Frisch, B. Hnich, Z. Kızıltan, I. Miguel, J. Pearson, and T. Walsh. Breaking row and column symmetries in matrix models. In P. Van Hentenryck, editor, *Proc. of CP'02*, vol. 2470 of *LNCS*, pages 462–476. Springer-Verlag, 2002.
8. P. Flener and J. Pearson. Breaking all the symmetries in matrix models: Results, conjectures, and directions. In *Proceedings of SymCon'02*, 2002. Available at `http://www.it.uu.se/research/group/astra/SymCon02/`.
9. K. Gilkes and M. Drexler. Drill-down approach for synthetic CDO Squared transactions. Standard and Poor's Publication, December 2003.
10. S. Jukna. *Extremal Combinatorics*. Springer-Verlag, 2001.
11. P. Meseguer and M. Sànchez. Tree-based Russian Doll Search: Preliminary results. In F. Rossi, editor, *Proceedings of the CP'00 Workshop on Soft Constraints*, 2000.
12. J.-F. Puget. Symmetry breaking using stabilizers. In F. Rossi, editor, *Proceedings of CP'03*, volume 2833 of *LNCS*, pages 585–599. Springer-Verlag, 2003.
13. G. Wetzel and F. Zabatta. A constraint programming approach to portfolio selection. In *Proceedings of ECAI'98*, pages 263–264, 1998.