∎

For 4SP, there is similar:

Theorem 4 *4SP(γ) is linear time solvable if $\gamma \geq 1/3$.*

Remarks

This paper discussed some subproblems of MPA. For other subproblems whose instances distribute in other manners, similar results may be developed. For example, small pools may be added into the inventory without affecting its major properties, differences between pool sizes may be made no less than some specific value. etc.

References

[1] S.J.Stolfo, L. Woodbury, J. Glazier, and P. Chan "The ALEXSYS Mortgage Pool Allocation Expert System: A Case Study of Speeding Up Rule-based Program", 1990.

[2] M. R. Garey and D. S. Johnson, "Computers and Intractability: a guide to the theory of NP-completeness", San Francisco: W. H. Freeman, 1979.

Map $A = \{a_1, a_2, \ldots, a_(2s+1)m\}$ to $P = \{p_1, p_2, \ldots, p_(2s+1)m\}$ and set $s(p_i) = s(a_i) + (2s/(2s+1))B$. So, $\sum_{p \in P} = m(2s+1)B$. Set $B' = (2s+1)B/s$. Then, A can be $(2s+1)$ partitioned to B if and only if P can be $(2s+1)$ partitioned to $(2s+1)B = sB'$. About the range of $s(p)$, we have

$$s(p)/B' = s(s(a) + (2s/(2s+1))B)/(2s+1)B = (s(2s+1)(s(a)/B) + 2s^2)/(2s+1)^2$$

because $1/2(s+1) < s(a) < 1/2s$, the above becomes

$$\frac{(s(2s+1)/2(s+1) + 2s^2)}{(2s+1)^2} < s(p)/B' < \frac{(s(2s+1)/2s + 2s^2)}{(2s+1)^2}$$

$$\frac{2s(2s+1) - s/(s+1)}{2(2s+1)^2} < s(p)/B' < \frac{2s(2s+1)+1}{2(2s+1)^2}$$

the left term is larger than c_1 in Lemma 1 and the right equals to c_2. Therefore $< P, s, B' >$ satisfies the conditions in the lemma, and P can be $(2s+1)$ partitioned into sB' if and only if it can be 3-split-partitioned into B'.

Hence, the mapping $< A, s, B, m > \rightarrow < P, s, B', n >$ described above is the desired reduction. ∎

For 4SP(γ), there are similar

Lemma 2 *In an instance of 4-SP,if $c_1 B < s(p) < c_2 B$ is true for all $p \in P$, where $c_1 = (3s(3s+1) - 1)/3(3s+1)^2$ and $c_2 = (3s(3s+1) + 1)/3(3s+1)^2$ for some positive integer s, then P can be 4-split-partitioned into B if and only if P can be $(3s+1)$-partitioned into sB.*

Theorem 2 *4SP(γ) is NP-complete for any $\gamma < 1/3$.*

5 Linear Time Solvable Instances

We have proved Problem 3SP(γ) is NP-complete if $\gamma < 1/2$. In this section we are going to give an interesting contrast to that result, as presented in the following theorem:

Theorem 3 *3SP(γ) is linear time solvable if $\gamma \geq 1/2$.*

Proof:
This can be done simply by putting all pools sequentially and then taking off amount B repeatively and sequentially, starting from the leftmost pool, the remain of a pool from filling one B will be used to fill the next.

No B can be formed with more than three pools, because otherwise, there would be two consecutive pools entirely, plus some parts from other pools, filled into B, which is impossible due to the condition $s(p) \geq \gamma B = 0.5B$

$$= 2v - 1 + 4s(u+v)/(2s(2s+1)+1) > 2v - 1.$$

So, the only possibility is $w = 2v$.

Now we can prove the validity of (2) by induction:

The first time when Step 5 is reached, $Block = \{G_i\}, u = 1$. Because exactly three components are needed to fill G_i, we have $|Comp(Block)| = 3 = 2u + 1$.

If (2) holds at some time when Step 5 begins, (2) will hold also. After this step is done, $|Block|$ becomes $u' = u+v$ and $Comp(Block)$ is expanded by $Comp(Newlevel) - Comp(Block)$ thus contains $2u + 1 + 2v = 2u' + 1$ elements. Then program goes back to Step 4, runs through it without further changing $Block$, forms a new $Newlevel$ of size v' and reaches Step 5 again. This time, (2) is still true except that u' is in place of u.

But Step 5 may not be done when $v = 0$. In this case, the current block is complete and to be committed to $Blockset$. The searching for next block will start as if previous blocks had not existed at all, therefore the above induction can repeat.

In a complete block, there are u groups, whose total volumn is uB, and all the amounts from the $2u+1$ components are fit into them. The following condition must be true:

$$(2u+1)(R-\epsilon) < uB < (2u+1)(R+\epsilon),$$

which can be transformed to

$$(R-\epsilon)/(B-2(R-\epsilon)) < u < (R+\epsilon)/(B-2(R+\epsilon)),$$

$$(2s(2s+1)-1)/(2(2s+1)^2-2(2s(2s+1)-1)) < u < (2s(2s+1)+1)/(2(2s+1)^2-2(2s(2s+1)+1))$$

$$s - (2s-1)/4(s+1) < u < s + (2s+1)/4s$$

its unique solution is $u = s$.

∎

Theorem 1 *3SP(γ) is NP-complete for any $\gamma < 1/2$.*

Proof:
For such a γ, there must be an integer s such that $\gamma < (2s(2s+1)-1)/2(2s+1)^2$ because the limit of the right hand side is $1/2$ when $s \to \infty$.

Now we are going to reduce the NP-complete problem (2s+1)-PARTITION to 3SP(γ) in polynomial time.

Given an instance of (2s+1)-PARTITION: A finite set A of $(2s+1)m$ elements, a bound $B \in Z^+$, and a "size" $s(a) \in Z^+$ for each $a \in A$, such that each $s(a)$ satisfies $1/((2s+1)+1) \le s(a)/B \le 1/((2s+1)-1)$ and such that $\sum_{a \in A} s(a) = mB$.

This instance also meets the requirements of instance in 3SP. Our idea is to change some quantities in the instance so that the sizes of the elements fit the restriction in Lemma 1 and partition is made equivalent to split-partition in that case.

1. Set $Blockset = \emptyset, NodesRamain = \{G_1, G_2, \ldots, G_n\}$.
2. If $NodesRamain = \emptyset$, exit. All the connective blocks of G are included in $Blockset$ in form of the node sets of the blocks.
3. Take any node from $NodesRamain$, assume it's G_i. Set $Block = \{G_i\}$.
4. Set $Newlevel = \{G_j \in NodesRamain | G_j$ is adjacent to a node in $Block$(that means,G_j has commo assume $Newlevel = \{G_{j1}, G_{j2}, \ldots, G_{jl}\}$.
5. if $Newlevel \neq \emptyset$, set $Block = Block \bigcup Newlevel\, NodesRamain = NodesRamain - Newlevel$, goto 4.
6. Otherwise, set $Blockset = Blockset \bigcup \{Block\}$, goto 3.

To prove each block contains exactly s nodes, we establish a few statements at the beginning of step 5:

Let

$$u = |Block|, \text{ the number of elements in } Block,$$

$$v = |Newlevel|, \text{ the number of elements in } Newlevel,$$

$$\epsilon = B/2(2s+1)^2, \text{ and } R = sB/(2s+1).$$

Then $c_1 B = R - \epsilon < s(p) < R + \epsilon = c_2 B$ for all $p \in P$. For any node set S, let

$$Comp(S) = \{p \in P | p \text{ is a component of nodes in } S.\}.$$

We claim (at the beginning of Step 5) that the following are true:

$$|Comp(Block)| = 2u + 1. \tag{2}$$

$$|Comp(Newlevel) - Comp(Block)| = 2v. \tag{3}$$

First we show (2) implies (3). Assume $|Comp(Newlevel) - Comp(Block)| = w$. For any group $G_j \in Newlevel$, at least one of its (no more than three) components must be in $Comp(Block)$. Hence, each G_j contributes at most two elements to $Comp(Newlevel) - Comp(Block)$. This fact leads to $w \leq 2v$.

On the other hand, the components in $Comp(Block) \bigcup Comp(Newlevel)$ should have a total amount large enough to cover $u + v$ groups, which can be represented as follows when (2) is true:

$$(2u + 1 + w)(R + \epsilon) > (u + v)B.$$

use $B = 2(2s+1)^2\epsilon$ and $R = 2s(2s+1)\epsilon$, we get

$$w > (u + v)B/(R + \epsilon) - 2u - 1$$

$$= (u + v)2(2s+1)^2/(2s(2s+1) + 1) - 2u - 1$$

QUESTION: Is there a matrix

$$SP = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ p_{k1} & p_{k2} & \cdots & p_{kn} \end{pmatrix}$$

such that

$$\text{for all } 1 \le i \le k, 1 \le j \le n, p_{ij} \text{ is a nonnegative integer,}$$

$$\sum_{j=0}^{n} p_{ij} = p_i, \ \sum_{i=0}^{k} p_{ij} = B,$$

$$\text{and there may be no more than 3 nonzero elements in a column.}$$

If the answer to the QUESTION is yes, we say that set P can be *3-split-partitioned into* B. Column j of matrix SP is called *group* j of this split-partition, element p_i is called a *component* of group j if $p_{ij} \neq 0$.

As we will see, how large the $s(p_i)$'s are is a key factor in determining the hardness of 3SP. For any nonnegative constant γ, we can define **Problem 3SP(γ)** as consisting of such instances of 3SP that $s(p) \ge \gamma B$ for all $p \in P$.

Problem 4SP and **4SP(γ)** are similar to the above.

Lemma 1 *In an instance of 3-SP,if* $c_1 B < s(p) < c_2 B$ *is true for all* $p \in P$, *where* $c_1 = (2s(2s+1)-1)/2(2s+1)^2$ *and* $c_2 = (2s(2s+1)+1)/2(2s+1)^2$ *for some positive integer* s, *then* P *can be 3-split-partitioned into* B *if and only if* P *can be* $(2s+1)$*-partitioned into* sB.

Proof: If P can be $(2s+1)$-partitioned into sB, take any partition group (it has $(2s+1)$ elements), without loss of generality, assume it contains $p_1, p_2, \ldots, p_{2s+1}$. Form subgroup $\{p_i, p_{i+1}\}$ for $i = 1, 3, 5, \ldots, (2s-1)$, totally s of them. Because the size of each element is less than $c_2 B < B/2$, the sum of the sizes of the two elements in a subgroup will be less than B. Then, make up these differneces with amounts from $s(p_{2s+1})$. This making up will consume exactly amount $s(p_{2s+1})$ because $\sum_{i=1}^{(2s+1)} p_i = sB$. Apply the above procedure to each partition group, we get a 3-split-partition of set P into B.

On the other hand, If P can be 3-split-partitioned into groups $G_1, G_2, \ldots, G_n$. First we represent the configuration with a graph $G =< V, E >$ where $V = \{G_1, G_2, \ldots, G_n\}$ and $E = \{(G_i, G_j) | G_i \text{ and } G_j \text{ have common components}\}$. We are going to prove that each connective block of graph G contains exactly s nodes, and there are totlaly $(2s+1)$ different components involved in. Therefore a $(2s+1)$ partition of set P into sB can be directly derived from those blocks since the component sets of connective blocks are pairwise disjoint.

Connective blocks can be separated out one by one by a simple procedure:

cision problem:

Problem 3DRMPA
INSTANCE: P, G, M, and $Tail$ the same as in 3RMPA. An integer thr to be the threshold value.
QUESTION: Is there an allocation matrix A such that all the conditions in 3RMPA's definition are satisfied except that the last one "$Profit(A)$ achieves maximal value" is changed to "$Profit(A) \geq thr$"?

4DRMPA's definition is similar.

A combinational optimization problem is NP-hard if and only if its corresponding decision problem is NP-complete. It has been shown in [2] that an NP-hard problem is polynomial time solvable if and only if **P=NP**.

4 NP-hardness of Mortgage Pool Allocation

The main results of this section are the NP-hardness of 3DRMPA and 4DRMPA.

It suffices to show the NP-hardness on subclasses of these problems. Here we obtain these subclasses by restricting the ranges of some quantities in their INSTANCE sets.

One kind of restriction is *High Extreme*, that is, let $Tail(g_i) = High$ for all i; Then, $Filled(g_i)$ should achieve its highest variance, i.e. $1.0249999M$, and the profit would be at the maximum $0.0249999nM$. We force this to be true by choosing $thr = 0.0249999nM$ in these subclasseses. Therefore, the inequality $Profit(A) \geq thr$ in their QUESTION parts becomes virtually a set of equations: $Filled(g_i) = 1.0249999M$ for $i = 1, 2, \ldots, n$.

Another restriction is *Low Extreme*, in which case all $Tail(g_i) = Low$, and again thr is chosen to be $0.0249999nM$. $Profit(A) \geq thr$ in QUESTION is equivalent to $Filled(g_i) = 0.9750001M$ for $i = 1, 2, \ldots, n$.

In both cases, $Tail$ is a constant mapping and thus can be omitted; M always appears with the variance constants, we can use another symbol B to denote $1.0249999M$ in High Extreme or $0.9750001M$ in Low Extreme. Further more, we can make the *total* amount in the inventory P just enough for the profit amount, that is, $\sum_{p_j \in P} Amount(p_j) = 1.0249999nM$ for High Extreme, or $\sum_{p_j \in P} Amount(p_j) = 0.9750001nM$ for Low Extreme. Hence, if QUESTION is answered positively, all pools will become empty after the allocation. With these constraints, we can describe 3DRMPA and 4DRMPA in a more concise but equvilent way:

Problem 3SP (3-Spilt-Partition)
INSTANCE: Nonnegative integer n (number of million-or-odd-lots) and B (amount to be filled to each of the n entries), a finite set (pool inventory) $P = \{p_1, p_2, \ldots, p_k\}$, and a nonnegative size(position of a pool) $s(p)$ for each $p \in P$, satisfying $\sum_{p \in P} s(p) = nB$.

3-PARTITION.
INSTANCE: A finite set A of $3m$ elements, a bound $B \in Z^+$, and a "size" $s(a) \in Z^+$ for each $a \in A$, such that each $s(a)$ satisfies $1/4 < s(a)/B < 1/2$ and such that $\sum_{a \in A} s(a) = mB$.
QUESTION: Can A be partitioned into m disjoint sets $S_1, S_2, \ldots, S_m$ such that, for $1 \leq i \leq m, \sum_{a \in S_i} s(a) = B$? (Notice that the above constraints on the item sizes imply that every such S_i must contain *exactly* three elements from A.)

It was proven in [2] that 3-PARTITION is a NP-complete problem in the strong sense.

We can define c-PARTITION problem in a similar way for any integer $c > 3$:

c-PARTIOTION
INSTANCE: A finite set A of cm elements, a bound $B \in Z^+$, and a "size" $s(a) \in Z^+$ for each $a \in A$, such that each $s(a)$ satisfies $1/(c+1) \leq s(a)/B \leq 1/(c-1)$ and such that $\sum_{a \in A} s(a) = mB$.
QUESTION: Can A be partitioned into m disjoint sets $S_1, S_2, \ldots, S_m$ such that, for $1 \leq i \leq m, \sum_{a \in S_i} s(a) = B$? (Notice that the above constraints on the item sizes imply that every such S_i must contain *exactly* c elements from A.)

If the answer to QUESTION is yes, we say A can be *c-partitioned into B*.

c-PARTITION can be proven NP-complete by constructing a polynomial reduction from 3-PARTITION. One way to do this is to add $(c-3)m$ elements of size B into set A in 3-PARTITION to form a new set A', and change the sum of the subsets from B to $B' = (c-2)B$. For A', we have $B'/4(c-2) \leq s(a) \leq B'/(c-2)$, and a c-partition exists if and only if in A a 3-partition exists. To make $s(a)/B'$ in proper range, we can increase each $s(a)$ by an amount x to become $s''(a) = s(a) + x$, then the subset sum will increase by cx, becoming $B'' = B' + cx$. Because $B'/4(c-2) \leq s(a) \leq B'/(c-2)$, we slove

$$\begin{cases} \frac{1}{(c+1)} \leq \frac{B'/4(c-2)+x}{B'+cx} \\ \frac{1}{(c-1)} \geq \frac{B'/(c-2)+x}{B'+cx} \end{cases}$$

and for $c > 4$ set $x = B'/(c-2)$, $s''(a)/B''$ will be in range $[1/(c+1), 1/(c-1)]$. For $c = 4$, we don't need this reduction because actually in [2], 4-PARTITION is first proven to be NP-complete and then reduce to 3-PARTITION.

Reader can notice that because $\lim_{x \to \infty} s''(a)/B'' = 1/c$, $s''(a)/B''$ can be adjusted into any range $[\alpha, \beta]$ as long as $\alpha < 1/c < \beta$.

Contrary to that of decision problem, the goal of combinational optimization problem is usually maximizing or minimizing a target amount T. In such a problem, however, if we keep the INSTANCE set the same except for adding a new threshold parameter thr, and change the goal to answering question "Is there any instance I such that $T(I) \geq thr$?" ($T(I) \leq thr$ for minization problem), then we get a decision problem corresponding to it.

For example, optimization problem 3RMPA is corresponding to the following de-

If we represent the allocation by a matrix:

$$A = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ p_{k1} & p_{k2} & \cdots & p_{kn} \end{pmatrix}$$

where every row corresponds to a pool and every column corresponds to a g_i, the above task can be expressed as: try to find a matrix A such that

$$\text{for all } 1 \leq i \leq k, 1 \leq j \leq n, \text{ we have:}$$

$$p_{ij} \geq 0 \text{ and } \sum_{j=0}^{n} p_{ij} \leq Amount(p_i);$$

$$0.975001M \leq Filled(g_j) = \sum_{i=0}^{k} p_{ij} \leq 1.024999M;$$

the number of non-zero elements in each column≤ 3, and $Profit(A)$ defined in (1) achieves maximal value.

The 3 in name 3RMPA is due to the restriction that no more than three pools can be used to fill a g_i (the number of non-zero elements in each of A's columns≤ 3). Therefore, **4RMPA** can be defined similarly.

It should be pointed out that we ignored some PSA restrictions for some security type in the above abstraction. For example, the 5K fragmentation rule, which means after the allocation, a pool should has either nothing remains or at least 5K remains; And the 5K spliting rule, which means amounts should be cut off a pool at points exactly corresponding to multiples of 5K in the original face of this pool. This complication, however, can be fotunately ruled out with the consideration that generally speaking, as the time passed, the 5K interval in the original face will have increasely narrowed counterpart in the current face, and finally becomes 1. On the other hand, even if we admitted the 5K interval, problem would simply become on the basis of integer set $\{0, 5K, 10K, 15K, \ldots\}$ instead of $\{0, 1, 2, 3 \ldots\}$, which is a matter of scale and would not influence the validity of our results. Anyway, remember that we are abstracting a subproblem from MPA and proving its hardness. If general forms of other restrictions are considered, needless to say, things will just become even harder.

In section 4, 3RMPA and 4RMPA will be shown to be NP-hard when the sizes of the pools satisfy some simple conditions. As a problem containing huge parameter instances of NP-hard problems, MPA can be considered intractable in practice.

3 A Few Results in NP-theory

Part of the contents in this section come from [2]. We describe problems in the same format as in that book, i.e., a decision problem consists of two parts: An instance set and a yes-no question. For example,

Because of this splitting, the real target of filling becomes the set of millions and odd lots $G = \{g_1, g_2, \ldots, g_n\}$ instead of the set of contracts C, again, we use $Amount(g_i)$ to denote g_i's amount. (In real world, this amount is less than or equal to 1,000,000, but in this paper, it is treated as unlimited.) By PSA rule, when filling g_i, a variance of $\pm 2.4999\%$ can be put on $Amount(g_i)$, that is, any amount between $Amount(g_i) \times 0.975001$ and $Amount(g_i) \times 1.024999$ is considered to be a legal amount to fill g_i (the multiplication, of course, should be understood with proper roundoff) . This is called 2.4999% *variance rule*, which, results in an opportunity of earning extra profit in the mortgage trade: If the price of the contract is higher than the current market price, larger amount should be filled and the most desired amount is the full positive variance $Amount(g_i) \times 1.024999$, contracts, millions or odd lots in this situation are "high tail"; Otherwise, smaller amount should be filled and the most desired is $Amount(g_i) \times 0.975001$, this situation is "low tail". If we use $Filled(g_i)$ to denote the amount actually filled to entry g_i in a allocation A, then the profit earning from high tails and money saved from low tails is (approximately) proportional to:

$$\sum_{g_i \text{ is high tail}} (Filled(g_i) - Amount(g_i)) + \sum_{g_i \text{ is low tail}} (Amount(g_i) - Filled(g_i)).$$

The difficulty of maximizing this quantity comes from, among other sources, the PSA restrictions on the number of pools being used in filling any single g_i. For example, for a g_i of GNMA security type, with coupon rate below 12%, if $Amount(g_i) = 1{,}000{,}000$, components coming from no more than three pools only can be used to fill it; If the coupon rate is 12% or more, no more than four pools can be used for this g_i. If $Amount(g_i)$ is less than 1,000,000, the restrictions are even more strict.

Now we can summarize the contract spliting, high/low tails, and pool number restrictions described above with a formal definition of a subproblem of MPA.

Problem 3RMPA(3-pool Retricted MPA)

Given pool set $P = \{p_1, p_2, \ldots, p_k\}$ and million-or-odd-lot set $G = \{g_1, g_2, \ldots, g_n\}$. Assume all g_i have the same amount, i.e. there exists an integer M such that $Amount(g_i) = M$ for all $1 \leq i \leq n$.

Given also a mapping $Tail: S \to \{High, Low\}$.

Try to find an allocation A in which each g_i is filled with components from no more than three pools in such a way that the sum of these components(i.e.$Filled(g_i)$) is within the 2.4999% variance of M, and the profit

$$Profit(A) = \sum_{Tail(g_i)=High} (Filled(g_i) - M) + \sum_{Tail(g_i)=Low} (M - Filled(g_i)). \quad (1)$$

is as large as possible.

1 Introduction

Mortgage pool allocation(MPA) is a procedure carried out periodically in financial institutions, during which large number of mortgage contracts are bought and sold within a single day. It is required that this procedure is organized in an efficient, accurate, reliable, and profitable way. This is a task beyond the capability of human allocators, therefore, a parallel export system approach is introduced in [1] to solve this problem. Here in this paper, we are going to give a justification for that approach by proving that problems like mortgage pool allocation are generally very hard, can not be solved very effectively by any fixed algorithm already known. As a powerful, flexible, and of high performance approach, the parallel export system supported by speeded compiler/interpretor is thus very reasonable to be used in the attacking of this kind of hard nuts.

2 Definition of the Problem

At the beginning of a MPA day, there is an initial inventory of mortgage pools in the bank, which consists of the remains from last MPA day. As the trade proceeds, amounts from the buying contracts are added up into the inventory while the selling contracts are filled with amounts from it. There is a whole set of PSA rules made by the government, together with some less rigid conventional agreements, to regulate this procedure. Within the extent of these rules and agreements, an allocation should be made with great efforts to reduce the number of sell contracts which can not be properly filled (called fails), and achieve highest profit.

More detailed description of the rules can be found in [1]. Here we give a basic introduction and derive a mathematical representation and abstraction suitable for the purpose of this paper. First, let's use $C = \{c_1, c_2, \ldots, c_m\}$ to denote the set of sell contracts, and $Amount(c_i)$ to represent the amount of contract c_i. Let $P = \{p_1, p_2, \ldots, p_k\}$ be the set (inventory) of pools, $Amount(p_j)$ represent the position (amount) of pool p_j. Because the buying part of MPA is passive and of little trick, it is not discussed here.

There are a few characteritics associated with contracts and pools, such as coupon rate, security type, etc. Pools can be used to fill a contract only if they have the same characteritics as the contract. PSA rules are slightly different for different characteristics. So, we can classify the contracts and the pools according to their characteristics, and do allocation separately for each class. Without loss of generality, we assume all elements of C and P have the same characteristics.

According to PSA rules, each contract c_i is first split into a number of whole millions and possibly an odd lot which is the remaindar of c_i devided by one million. To fill the contract is actually to fill each of its whole millions or odd lot with amounts from a specific number of pools, and then the properly filled millions (called good million or GM) and the odd lot are merged back to form a properly filled contract c_i.

Complexity of Mortgage Pool Allocation*

Salvatore J. Stolfo *Xiangdong Yu*

Department of Computer Science
Columbia University
New York, NY 10027

and
Center for Advanced Technology
New York, NY 10027

September 30, 1990

Abstract

A new class of NP-hard problems are proved and this result is applied to the complexity analysis of the mortgage pool allocation problem in American banks. A boundary on pool sizes is shown to sharply distinguish hard instances from easy ones in this kind of combinational optimazition problems.

*Technical Report Number: CUCS-063-90