Optim Lett (2014) 8:1183–1190
DOI 10.1007/s11590-013-0636-3

SHORT COMMUNICATION

Algorithm for quadratic semi-assignment problem with partition size coefficients

Maciej Drwal

Received: 17 July 2012 / Accepted: 15 March 2013 / Published online: 27 March 2013

Abstract This paper discusses the problem of assigning N streams of requests (clients) to M related server machines with the objective to minimize the sum of worst-case processing times. The completion time of a batch of requests is measured as a sum of weights of the subset of clients which share a single machine. Such problem can be seen as minimizing the sum of total weights of blocks of M-partition, each multiplied by the cardinality of a block. We prove that this problem can be solved in polynomial time for any fixed M and present an efficient backward induction algorithm.

Keywords Binary programming · Set partition · Exact algorithms

1 Introduction

Consider the following problem of assigning requests to processing servers in distributed computing system. We are given a set of M server machines, each with processing speed $1/h_j$, $j = 1, \ldots, M$, and a set of N clients, each with a weight w_i, $i = 1, \ldots, N$. Parameter h_j is the time of single request processing on jth server. A weight w_i represents the amount of client's demand for processing (number of requests sent in a unit of time), where client itself can be interpreted either as a single application or whole local area network, issuing a continuous stream of requests. The processing requirement after assigning ith client to jth machine is $c_{ij} = w_i h_j$. However, when multiple clients share the same machine, the order of request processing is unknown, thus each client is interested in minimal worst-case processing time

M. Drwal (✉)
Institute of Computer Science, Wroclaw University of Technology,
Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland
e-mail: maciej.drwal@pwr.wroc.pl

(e.g. due to the quality of service requirements). Hence the waiting (completion) time of ith client's batch of requests is defined as $C_i = \sum_{j=1}^{M} x_{ij} \sum_{k=1}^{N} x_{kj} c_{kj}$, where $x_{ij} = 1$ if ith client is assigned to jth server, and $x_{ij} = 0$ otherwise. It is assumed that demand of each client is unsplittable.

The objective is to assign all clients in such a way so as to minimize the sum of completion times, $\sum_{i=1}^{N} C_i$. The quadratic integer programming formulation is:

$$\text{minimize} \sum_{i=1}^{N} \sum_{j=1}^{M} \left(h_j x_{ij} \sum_{k=1}^{N} w_k x_{kj} \right) = \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{N} c_{kj} x_{ij} x_{kj} \tag{1}$$

subject to:

$$\forall_i \sum_{j=1}^{M} x_{ij} = 1, \tag{2}$$

$$\forall_{i,j} x_{ij} \in \{0, 1\}. \tag{3}$$

This problem can be seen as a special case of quadratic semi-assignment problem (QSAP) [10,11], which is well-known to be NP-hard [1], and for which optimal solutions are difficult to compute even for small-sized instances. QSAP is obtained by relaxing the one-to-one constraint in generalized quadratic assignment problem [7]. One specific class of polynomially solvable instances of QSAP has been characterized in [9], where its applications in flight scheduling are discussed. In this paper another class of QSAP instances of practical importance is identified, which also can be solved efficiently, under certain fixed-parameter assumption. In the presented formulation the quadratic terms reflect the increased delays caused by multiple clients competing for single processor [8]. Similar load balancing problems were of the interest in the context of selfish resource allocation in the Internet, where each assigned task experiences the machine completion time [5]. Other applications of the presented problem include specialized variants of correlation clustering [2] and scheduling [6].

Since the instance size of this problem is characterized by two parameters, N and M, in this paper we show fixed-parameter tractability of (1)–(3). The motivation is based on the observation that in practical applications the number of servers M is usually much smaller than the number of clients N needed to be assigned. In general, when also M is considered as a part of the input, problem (1)–(3) is NP-hard [3].

2 Main results

In this paper we introduce the following equivalent formulation of (1)–(3) to develop the analysis. Given a multiset $U = \{w_1, \ldots, w_N\}$ of positive integers (weights), find such a partition of weights $S_1, S_2, \ldots, S_M$, that minimizes function:

$$f(S_1, S_2, \ldots, S_M) = \sum_{j=1}^{M} \left(|S_j| h_j \sum_{w_i \in S_j} w_i \right) \tag{4}$$

where $\bigcup_j S_j = U,\ S_i \cap S_j = \emptyset$, for all $i,\ j = 1, \ldots, M,\ i \neq j$. In this reformulation of semi-assignment, the partition block sizes $|S_j|$ and reciprocals of speeds h_j assume the role of coefficients. Each S_j represents a server. The presented technique is based on the following simple observation:

Proposition 1 *For $M = 2$ the problem (1)–(3) can be solved in polynomial time.*

Proof For two machines the problem can be seen as follows. There is a multiset of weights $U = \{w_1, w_2, \ldots, w_N\}$. We need to partition the set of indices $T = \{1, 2, \ldots, N\}$ into two disjoint subsets, S and $T - S$, in such a way that the set-function:

$$f(S) = h_1|S| \sum_{i \in S} w_i + h_2(N - |S|) \sum_{i \in T-S} w_i$$

is minimized.

Denote $C = \sum_{i=1}^{N} w_i$. Let us consider the following family of one-variable linear functions:

$$g_k(x) = h_1 kx + h_2(N - k)(C - x) = ((h_1 + h_2)k - h_2 N)x + h_2(N - k)C.$$

Clearly, each g_k coincides with $f(\cdot)$ on arguments $x = \sum_{i \in S} w_i$, where S belongs to subset of f's domain given by $\{S \subset T : k = |S|\}$, for fixed k. Thus f can be written as:

$$f(S) = \begin{cases} h_2 NC, & S = \emptyset, \\ h_1 NC, & S = T, \\ g_k(x), & x \in A_k, \quad \text{for } k = |S| = 1, \ldots, N-1, \end{cases}$$

where

$$A_k = \left\{ x = \sum_{i \in S} w_i : S \subset T, \quad k = |S| \right\}.$$

If $\frac{d}{dx} g_k(x) = ((h_1 + h_2)k - h_2 N) \geq 0$ then g_k attains minimum for the smallest $x \in A_k$, and otherwise for the largest $x \in A_k$. Such x can be computed by sorting elements of A_k and summing k first weights [k last, in case of $((h_1+h_2)k-h_2N) < 0$]. Thus minimum of f can be found by computing $g_k(\min A_k)$ or $g_k(\max A_k)$ for all $k = 1, \ldots, N-1$. Since sorting requires $O(N \log N)$ time, and precomputing all partial sums $w_1 + \cdots + w_k$ can be accomplished in $O(N)$ time, thus finding the minimum (or maximum) of g_k can be performed in constant time for every k. The overall complexity is $O(N \log N)$. □

The above argument shows that the size of the set of feasible solutions, which is exponential in N, can be easily reduced, by leaving only polynomially many potential solutions to check. It turns out that this technique can be generalized. For any fixed $M \in \mathbb{N}$ the number of integer solutions of the equation $k_1 + k_2 + \cdots + k_M = N$ is

bounded by $O(N^{M-1})$ [the asymptotics is in fact $N^{M-1}/M!(M-1)!$ see [4] for more details on counting integer partitions].

The idea is to search for the solutions on properly constructed hyperplanes in $\mathbb{R}^m$. Given a hyperplane $\sum_{i=1}^m \alpha_i x_i = \gamma$, we can determine its orientation by analyzing signs of coefficients α_i. For positive α_i the corresponding coordinate x_i should be as small as possible, which amounts to taking the sum of smallest weights, since the hyperplane-defining function increases in these directions. On the other hand, for negative α_i the corresponding x_i should be as large as possible, which amounts to taking the sum of largest weights, since the hyperplane-defining function decreases in these directions. Utilizing the fact that the sum of coordinates $\sum_{i=1}^m x_i$ must be equal to the total weight of clients in U, we can reduce the problem of searching for values of $\mathbf{x} = (x_1, \ldots, x_m)$ on hyperplane in $\mathbb{R}^m$ to a smaller problem of searching on a hyperplane in $\mathbb{R}^{m-1}$, by expressing a variable x_j in terms of other coordinates. Starting from $m = M$, and repeating until $m = 1$, we obtain an algorithm which runs in time polynomial in $N = |U|$.

Theorem 1 *For any fixed $M \in \mathbb{N}$ the problem (1)–(3) can be solved in polynomial time.*

Proof Since the sets $S_1, \ldots, S_M$ are disjoint and must cover exactly N elements w_i, there are polynomially many possible combinations of cardinalities of sets $S_1, \ldots, S_M$. It is enough to consider one fixed choice of partition blocks sizes. Let us fix $k_j = |S_j|$, for all $j = 1, \ldots, M$. Observe that function (4) satisfies:

$$\begin{aligned}&\min\{f(S_1, \ldots, S_M) : |S_j| = k_j, j = 1, \ldots, M\}\\ &\quad = \min\left\{ g(\mathbf{x}) = \sum_{j=1}^{M} h_j k_j x_j : x_j = \sum_{w_i \in S_j} w_i, |S_j| = k_j, j = 1, \ldots, M \right\}.\end{aligned}$$

Consider a hyperplane given by equation $g(\mathbf{x}) = \sum_{j=1}^m \alpha_j x_j - \gamma$. Let us separate the indices of variables $\mathbf{x}$ into two disjoint sets A^+ and A^-:

$$A^+ = \left\{ j : \frac{\partial}{\partial x_j} g(\mathbf{x}) \geq 0 \right\}$$

and $A^- = \{1, \ldots, m\} \backslash A^+$. Observe that the minimum of (4) lies on the hyperplane $g(\mathbf{x})$, with $m = M$, $\alpha_j = h_j k_j$ and $\gamma = 0$, for some $\mathbf{x}$ satisfying $\sum_{i=1}^M x_i = \sum_{w \in U} w$. Clearly, since in directions x_j for $j \in A^+$ the function g is nondecreasing, an optimal $\mathbf{x}$ must attain the smallest possible values on these coordinates. Similarly, for optimal $\mathbf{x}$ the coordinates x_j for $j \in A^-$ should be as large as possible, since g decreases in those directions. Denote $k_{\min} = \sum_{j \in A^+} k_j$ and $k_{\max} = \sum_{j \in A^-} k_j$. Without the loss of generality, let $w_1 \leq w_2 \leq \cdots \leq w_N$. Since $k_{\min} + k_{\max} = N$, thus coordinates $j \in A^+$ should sum up to the value of sum of exactly $k_{\min}$ smallest weights:

$$\sum_{j \in A^+} x_j = \sum_{i=1}^{k_{\min}} w_i, \tag{5}$$

and the remaining coordinates should sum up to the value of sum of $k_{\max}$ largest weights:

$$\sum_{j\in A^-} x_j = \sum_{i=N-k_{\max}+1}^{N} w_i. \tag{6}$$

Let x_{j^+} be any variable such that $j^+ \in A^+$, and let x_{j^-} be any variable such that $j^- \in A^-$ (at least one of these sets must be nonempty). From (5) and (6) we get:

$$x_{j^+} = \sum_{i=1}^{k_{\min}} w_i - \sum_{i\in A^+\setminus\{j^+\}} x_i \tag{7}$$

and:

$$x_{j^-} = \sum_{i=N-k_{\max}+1}^{N} w_i - \sum_{i\in A^+\setminus\{j^-\}} x_i. \tag{8}$$

Substituting x_{j^+} and x_{j^-} into the hyperplane equation $g(\mathbf{x})$ we obtain an equivalent function $\hat{g}(\hat{\mathbf{x}})$ of $m-2$ variables (or $m-1$ variables, in case if one of the sets A^+ and A^- was empty):

$$\begin{aligned}\hat{g}(\hat{\mathbf{x}}) = &\sum_{j\in A^+\setminus\{j^+\}} (\alpha_j - \alpha_{j^+})x_j + \sum_{j\in A^-\setminus\{j^-\}} (\alpha_j - \alpha_{j^-})x_j \\ &+\alpha_{j^+} \sum_{i=1}^{k_{\min}} w_i + \alpha_{j^-} \sum_{i=N-k_{\max}+1}^{N} w_i.\end{aligned} \tag{9}$$

This implies that, assuming optimal $\hat{\mathbf{x}}$ is known, remaining variables of optimal $\mathbf{x}$, namely x_{j^+} and x_{j^-}, can be computed from relations (7) and (8), by backward induction.

Observe that the above reasoning can be applied again to the obtained hyperplane $\hat{g}(\hat{\mathbf{x}}) = \sum_{j=1}^{m'} \hat{\alpha}_j x_j - \hat{\gamma}$, with parameters $\hat{\alpha}_j$ and $\hat{\gamma}$ defined as in Eq. (9). Moreover, the choice of weights is now narrowed for all coordinates x_j with $\alpha_j \geq 0$ to the subset of $k_{\min} - k_{j^+}$ first elements of the sorted sequence of weights, and for all coordinates x_j with $\alpha_j < 0$ to the subset of $k_{\max} - k_{j^-}$ its last elements. Thus the set of remaining weights to allocate has now $N - k_{j^+} - k_{j^-}$ elements.

After at most $M-1$ steps we obtain a 1-dimensional hyperplane $\hat{g}(x) = \hat{\alpha}_1 x - \hat{\gamma}$, which allows computing the final variable x by inspection, as in the proof of Theorem 1. The whole process can be accomplished in $\Omega(N \log N) + O(MN)$ steps [for each of $O(N^{M-1})$ choices of partition block sizes]. □

Algorithm 1 implements the presented idea. To solve the problem (1)–(3) it is enough to run Algorithm 1 with arguments $(U, M, h_1 k_1, \ldots, h_M k_M, 0)$ subsequently

Algorithm 1 Procedure for finding optimal partition for given block sizes.

Require: $W = \{w_1, \ldots, w_N\}$ (sorted multiset of weights, $w_1 \leq \ldots \leq w_N$), m (number of unknown blocks in partition), $k_1, \ldots, k_m$ (requested sizes of partition blocks), $\alpha_1, \ldots, \alpha_m$, γ (coefficients of hyperplane in $\mathbb{R}^m$)

Ensure: $X_1, \ldots, X_m$ (m blocks of computed partition)

1: **function** SOLVE($W, m, k_1, \ldots, k_m, \alpha_1, \ldots, \alpha_m, \gamma$)
2: **if** $m = 1$ **then**
3: **return** W
4: **end if**
5: $n \leftarrow \arg\min\{\alpha_i \ : \ i = 1, \ldots, m\}$
6: **for** $j = 1, \ldots, m$ **do**
7: $\alpha'_j \leftarrow \alpha_j - \alpha_n$
8: **end for**
9: $k_{\min} \leftarrow \sum_{i=1}^{m} k_i - k_n$
10: $\gamma' \leftarrow \gamma + \alpha_n \sum_{i=1}^{k_{\min}} w_i$
11: $W' \leftarrow \{w_1, \ \ldots, \ w_{k_{\min}}\}$
12: $(X_1, \ldots, X_{n-1}, X_{n+1}, \ldots, X_m) \leftarrow$
13: SOLVE($W', m-1, k_1, \ldots, k_{n-1}, k_{n+1}, \ldots, k_m, \alpha'_1, \ldots, \alpha'_{n-1}, \alpha'_{n+1}, \ldots, \alpha'_m, \gamma'$)
14: **return** $(X_1, \ldots, X_{n-1}, W \backslash W', X_{n+1}, \ldots, X_m)$
15: **end function**

for all allowed partition sizes $k_1, \ldots, k_M$. Correctness of the routine follows from the following fact:

Proposition 2 *Given an instance of problem* (1)–(3) *and a sequence of integers* $k_1, \ldots, k_M$*, Algorithm 1 computes* M*-partition of set* $\{w_1, w_2, \ldots, w_N\}$*, such that each* n*th block has* k_n *elements, and the cost* (1) *is minimal.*

Proof Suppose $m \geq 2$, as when $m = 1$ a trivial 1-partition $S_1 = W$ is returned. Observe that initially all parameters $\alpha_j = h_j k_j$ are nonnegative. In step 5 the Algorithm 1 finds such n that α_n is minimal. This guarantees that in step 7 all resulting parameters α'_j remain nonnegative. These parameters correspond to $\hat{\alpha}_j = \left(\alpha_j - \alpha_{j^+}\right)$ in (9). Consequently, $A^- = \emptyset$ and only set A^+ is nonempty in each recursive call (step 13). Thus the current hyperplane is always defined exclusively by nonnegative parameters $\hat{\alpha}_j$ and $\hat{\gamma}$.

Due to this, since given m-dimensional hyperplane determines the allocation of $\sum_{i=1}^{m} k_i$ smallest weights, the next hyperplane (of dimension $m-1$) determines the allocation of $\sum_{i=1}^{m} k_i - k_n$ smallest weights, as k_n largest of them correspond to the partition block $S_n = W \backslash W'$, where W' is as given in step 11. The contents of all but nth block is determined by calling the procedure recursively for the subsequent hyperplane. This results in computing $(m-1)$-partition of set W', until $m = 1$. In step 14, after returning from the recursive call, the final partition vector is updated by inserting S_n. □

As an example, consider five servers with parameters $h_j = 2, 1, 5, 3, 1$, respectively, and five clients with demands $w_i = 5, 3, 1, 2, 2$, respectively. The optimal partition block sizes are 1, 2, 0, 1, 1. The algorithm allocates total demand 13 to servers 1, 2, 4 and 5. In the second iteration variable x_5 is eliminated, leaving total demand 8 to allocate among servers 1, 2, 4. Next, x_1 is eliminated, leaving demand 5 for servers 2

Table 1 Solutions of example problem instances computed by Algorithm 1 along with their respective running times

N	(M)	Partition	Solution value	Running time (s)
5	8	(1, 1, 1, 2)	1297.6	2.2
5	9	(1, 1, 1, 2)	921.6	23.3
5	10	(1, 1, 2, 1)	1,107.1	441.0
30	6	(4, 8, 5, 5, 4, 4)	38,024.3	57.8
30	7	(5, 7, 4, 4, 4, 3, 3)	32,427.1	438.1
30	8	(3, 5, 3, 2, 3, 3, 2, 9)	19,895.1	3,016.9
50	2	(27, 23)	381,107.0	1.13
50	3	(18, 16, 16)	223,302.9	1.25
50	4	(12, 14, 11, 13)	163,382.3	4.52
50	5	(9, 13, 10, 10, 8)	139,832.9	61.1
75	2	(41, 34)	883,630.5	57.9
75	3	(28, 23, 24)	512,673.5	56.6
75	4	(23, 18, 17, 17)	375,971.1	70.6
75	5	(14, 17, 16, 18, 10)	321,947.7	481.4
100	2	(54, 46)	14,89,909.4	1,544.1
100	3	(38, 31, 31)	865,279.5	1,516.5
100	4	(21, 26, 31, 22)	635,263.0	1,558.1
100	5	(20, 23, 19, 22, 16)	539,244.1	3,308.6

and 4. Finally, x_2 is eliminated. This results in optimal solution {3}, {2, 2}, ∅, {1}, {5}, with value 22.

3 Computational experiments

An experimental study has been carried out in order to evaluate the efficiency of Algorithm 1 in practice. The most time-consuming part in computing optimal solution is the searching among all feasible partition sizes $k_1, k_2, \ldots, k_M$, since their number is bounded by a polynomial of degree $M - 1$, while the routine which computes the partition itself has complexity bounded by $O(N \log N)$. For relatively small M the algorithm is very fast, but its performance drops dramatically with the increase of M. This is illustrated in Table 1, where example solutions are presented, along with corresponding partition sizes and respective running times (in seconds). The instances were randomly generated by drawing $w_i \in (0, 100)$ and $h_j \in (0, 10)$ from uniform distributions.

4 Conclusions

The class of quadratic semi-assignment problem instances presented in this paper models the task of assigning streams of requests to a group of related server machines.

In the presence of unknown exact request schedules, the objective is to minimize the sum of the worst-case processing times, which are equal for a subset of clients sharing the same machine. Although the corresponding generalization is NP-hard, in practical applications the number of servers is usually much smaller than the number of clients to assign. Thus especially important is the case with fixed number of servers, for which exact polynomial time algorithm was given. The demonstrated reformulation of the problem, minimizing sum of partition blocks with block size-dependent weights may also be useful in the design of algorithms for other combinatorial problems involving operations on set partitions.

Acknowledgments This research is supported by the scholarship co-financed by European Union within European Social Fund.

References

1. Burkard, R., Cela, E., Pardalos, P., Pitsoulis, L.: The quadratic assignment problem. In: Handbook of Combinatorial Optimization. Kluwer Academic Publishing, New York (1998)
2. Demaine, E., Emanuel, D., Fiat, A., Immorlica, N.: Correlation clustering in general weighted graphs. Theor. Comput. Sci. **361**(2), 172–187 (2006)
3. Farahani, R., Hekmatfar, M.: Facility Location: Concepts, Models, Algorithms and Case Studies. Physica-Verlag, Heidelberg (2009)
4. Flajolet, P., Sedgewick, R.: Analytic Combinatorics. Cambridge University Press, London (2009)
5. Koutsoupias, E., Papadimitriou, C.: Worst-case equilibria. In: Proceedings of the 16th Annual Conference on Theoretical Aspects of Computer Science, pp. 404–413. Springer, Berlin (1999)
6. Lawler, E., Lenstra, J., Rinnooy Kan, A., Shmoys, D.: Sequencing and scheduling: Algorithms and complexity. In: Handbooks in Operations Research and Management Science, vol. 4, pp. 445–522 (1993)
7. Lee, C., Ma, Z.: The generalized quadratic assignment problem. Research Report, Department of Mechanical and Industrial Engineering, University of Toronto, Canada (2004)
8. Magirou, V., Milis, J.: An algorithm for the multiprocessor assignment problem. Oper. Res. Lett. **8**(6), 351–356 (1989)
9. Malucelli, F.: A polynomially solvable class of quadratic semi-assignment problems. Eur. J. Oper. Res. **91**(3), 619–622 (1996)
10. Malucelli, F., Pretolani, D.: Lower bounds for the quadratic semi-assignment problem. Eur. J. Oper. Res. **83**(2), 365–375 (1995)
11. Pentico, D.: Assignment problems: a golden anniversary survey. Eur. J. Oper. Res. **176**(2), 774–793 (2007)