Local Linear Time Convergence of Primal-Dual Energy Minimization Algorithm for Parallel Processing

Håkan Lennerstad*

*Department of Mathematics and Natural Science, Blekinge Insitute of Technology, Sweden

Abstract—We consider energy minimization by speed-scaling of an open shop multiprocessor with n jobs and m machines. The paper studies the complexity of a primal-dual solution algorithm of [4], which was an open question in that paper. We prove that in a neighbourhood of the solution the complexity of the algorithm is $O(mn \log \frac{1}{\varepsilon})$ if $n \neq m$ and ε is the roundoff error of the computer.

The paper demonstrates how linearization can be used to investigate the complexity of an algorithm close to the optimum. An estimate of the size of the neighbourhood where the linearization error is smaller than the computer's roundoff error is also given.

I. INTRODUCTION

Multiprocessor scheduling involving energy minimization is a rapidly emerging research area. Energy minimization can be done in many different ways. In [3] the author considers feasibility together with energy-awareness in multiprocessor scheduling of periodic tasks with identical processors. The processors have variable speed, and the power consumption as a function of the speed is assumed to be increasing and strictly convex. Here energy is minimized among all feasile schedules, which is proven to be an NP-hard problem in the strong sense. In [5] a review over state-of-the-art energy-efficient scheduling on dynamic voltage scaling platforms is provided. Heuristics for the same platform are developed in [11], among other things slack in non-critical processors are energy reduced by lowering the speed. In [9], also considering a dynamic voltage scaling platform, a goal function is defined that performs a trade-off between quality of schedules and energy consumption. Overviews over recent research in speed-scaling can be found in [1] and [2].

In [6] a polynomial time algorithm is provided that minimizes the makespan of an open-shop scheduling problem. This is close to the present paper, which concerns open-shop speed-scaling where energy is minimized by adjusting the speeds for the processors.

Related work. The primal-dual interior point method for convex programming has a long history; an overview is provided in [12]. It has recently been extensively used for various applications such as queing networks, economics and image analysis. In [8], Kuhn presented the so called Hungarian method, which is a primal-dual algorithm for a matching problem, later modified in [7] to find market equilibrium when utilities for goods are linear. The combinatorial algorithm of [7] is proven to be of polynomial time complexity, which is

the first such algorithm. It is not strongly polynomial, which however is achieved by the algorithm in [10] by using a different termination criterion.

Multiprocessor scheduling and primal-dual solution algorithms come together in the paper [4]. Here power consumtion is, as is common, assumed to be described by the function s^α for a processor speed s, where $\alpha > 1$ is a machine dependent constant. This allows an explicit convex optimization problem to be formulated.

Problem formulation. The speed-scaling preemptive open-shop problem involves a set of n jobs $\mathcal{J} = \{J_1, , ..., J_n\}$ and a set of m processors $\mathcal{M} = \{M_1, , ..., M_m\}$. Each job consists of operations that have to run on different processors. Operations of the same job cannot be executed simultaneously. The operation O_{ij} of the job J_j has to be executed on processor M_i and has an amount of work $w_{ij} \geq 0$. So $w_{ij} = 0$ is allowed, in which case there is no operation for that pair (i, j). We will use the notation $O_{ij} \in J_j$ to denote that operation O_{ij} belongs to J_j for a certain j, and $O_{ij} \in M_i$ denotes that O_{ij} is to be executed on processor M_i. The operations may be preempted, i.e. interrupted and continued later. The goal is to minimize the total energy consumed such that all operations are completed before a given deadline d. In [4] this energy minimizing scheduling problem is formulated as a non-linear optimization problem. A primal-dual solution algorithm is presented, and the convergence to the unique solution is proven. However, the complexity of the algorithm is not known, which is formulated as an open problem in [4].

Contribution of this paper. The present paper provides a partial answer to the open problem of [4] by establishing local linear convergence in both n and m if $n \neq m$, i.e. the algorithm has linear complexity in n and in m separately in some neighbourhood of the solution (Theorem 1). This is in well accordance with experiments presented in [4]. For $n = m$ we have no complexity result, and experiments show very slow or no convergence in this case. An estimate of the size of the neighbourhood is also provided, as well as estimates of the error due to the linearization outside this neighbourhood.

II. THE CONVEX PROBLEM AND THE PRIMAL-DUAL ALGORITHM

A. The convex problem

If operation O_{ij} has speed s_{ij} and work w_{ij}, the completion time is $\frac{w_{ij}}{s_{ij}}$ and the power s_{ij}^α means that the corresponding energy required, to be minimized, is $\frac{w_{ij}}{s_{ij}}s_{ij}^\alpha = w_{ij}s_{ij}^{\alpha-1}$. The minimization problem in the variables s_{ij} is then

$$\min \sum_{i:O_{ij}\in J_j}\sum_{j:O_{ij}\in M_i} w_{ij}s_{ij}^{\alpha-1}$$

$$\sum_{j:O_{ij}\in M_i} w_{ij}s_{ij}^{-1} \leq d, \qquad 1 \leq i \leq m$$

$$\sum_{i:O_{ij}\in J_j} w_{ij}s_{ij}^{-1} \leq d, \qquad 1 \leq j \leq n$$

$$s_{ij} \geq 0, \qquad O_{ij}\in J_j, O_{ij}\in M_i.$$

which is a non-linear and convex minimization problem.

For such problems, the Karush-Kuhn-Tucker conditions provide a solution. We thus introduce Lagrange multipliers β_i, $1 \leq i \leq m$, for the conditions (1), and γ_j, $1 \leq j \leq n$, for the conditions (1). In [4] is is shown that the stationarity conditions imply that

$$s_{ij}^\alpha = \frac{\beta_i + \gamma_j}{\alpha - 1}, O_{ij}\in J_j, O_{ij}\in M_i. \tag{1}$$

It is typical for a non-linear problem to obtain a direct relationship between primal and dual variables. This allows us to eliminate the primal variables, and find a solution only in terms of the dual variables. Intuitivelty, each β_i denotes the contribution of processor M_i to the speed of corresponding operations, and γ_j is the contribution from the job J_j.

B. The complementary problem

The problem can now be defined entirely in terms of the β_i's and the γ_j's since it turns out that the dual variables corrisponding to the inequalities $s_{ij} \geq 0$ are all zero. Using (1) to eliminate s_{ij}, the complementary slackness conditions becomes

$$\beta_i \left(\sum_{O_{ij}\in M_i} \frac{w_{ij}}{\sqrt[\alpha]{\frac{\beta_i+\gamma_j}{\alpha-1}}} - d \right) = 0, \qquad 1 \leq i \leq m \tag{2}$$

$$\gamma_j \left(\sum_{O_{ij}\in J_j} \frac{w_{ij}}{\sqrt[\alpha]{\frac{\beta_i+\gamma_j}{\alpha-1}}} - d \right) = 0, \qquad 1 \leq j \leq n, \tag{3}$$

which together with the feasibility conditions

$$\sum_{O_{ij}\in M_i} \frac{w_{ij}}{\sqrt[\alpha]{\frac{\beta_i+\gamma_j}{\alpha-1}}} \leq d, \qquad 1 \leq i \leq m$$

$$\sum_{O_{ij}\in J_j} \frac{w_{ij}}{\sqrt[\alpha]{\frac{\beta_i+\gamma_j}{\alpha-1}}} \leq d, \qquad 1 \leq j \leq n.$$

define the complementary problem. Any solution in β_i's and γ_j's to this problem gives by (1) the unique solution in the variables s_{ij}. Note that different solutions in β_i's and γ_j's may give the same s_{ij}-solution.

C. The primal-dual algorithm

The primal-dual algorithm by [4] solves the complementary problem by starting with all $\beta_i = 0$ and all γ_j given the maximal value $(\alpha-1)\left(\sum_{O_{ij}\in M_i} w_{ij}/d\right)^\alpha$. This is obtained from (1) with $\beta_i = 0$, and for each γ_j the corresponding feasibility condition (4).

With this initialization one can expect some processors to be infeasible, i.e. some of the 2-conditions are violated. Then the next step of the algorithm is to *increase* the β_i's corresponding to infeasible conditions so also these processors become feasible, and at the limit of being unfeasible.

After this step some machines may not be tight, i.e. some of the 3-conditions are violated. So the next step is to *decrease* all γ_j's corresponding to untight machines, except all γ_j's that are zero. Thus these become tight γ_j's at the border of untight.

These two steps constitute one round of the algorithm. They are repeated until there are no unfeasible processors, and all machines with $\gamma_j > 0$ are tight, i.e. the error is smaller than the round-off error of the computer. The convergence is proved in [4].

As described, the rate of convergence of this primal-dual algorithm is the focus of the present report. The experimental results presented in [4] show rapid convergence for $m \neq n$, and no large change in convergence rate as we come closer to the optimum. If $m = n$, experiments have shown either very slow convergence, or the computation has been cancelled due to long processing time. These experimental observations are both strongly in line with the results in this paper.

III. MAIN RESULTS

Suppose that $\{B_i\}_{i=1}^m$ and $\{\Gamma_j\}_{j=1}^n$ is the solution to the problem. Denote by $S_{ij} = \left(\frac{B_i+\Gamma_j}{\alpha-1}\right)^{1/\alpha}$ the optimal speed of each operation and by $\Delta s_{ij} = (s_{ij}^\alpha - S_{ij}^\alpha)^{\frac{1}{\alpha}}$ the deviation of the speed from the optimal as the algorithm searches the optimal speed. We also denote the minimum speed by $S_{\min} = \min_{i,j}\{S_{ij} : S_{ij} > 0\}$. Suppose that the roundoff error in the computer is δ, and that the maximum error for accepting a point as a solution is ε. It is natural to assume $\varepsilon = \delta$.

The following theorem is the main result of the paper.

Theorem 1: Denote $\Delta s_{\min} = \delta^{\frac{1}{2\alpha}} S_{\min}^{\frac{1}{2\alpha^2}+\frac{1}{\alpha}}$ and suppose that $m \neq n$. Then if all $|\Delta s_{ij}| \leq \Delta s_{\min}$, the number of floating point operations needed to reach a solution is $O(mn \ln \frac{1}{\varepsilon})$.

The theorem is proven in Section III-C. In that section the size of a neighborhood in which there is no difference between the original and the linearized problem is provided (Lemma 4). Section III-A linearizes the problem around the (unknown) solution, and in Section III-B the complexity of the linearized problem is found to be $O(mn \ln \frac{1}{\varepsilon})$ (Theorem 3).

Experiments imply that the rate of convergence is not much different outside the neighbourhood [4]. The rate of convergence in the therorem is a strict, pessimistic and provable result. It is an open question whether the non-linearity makes the convergence slower or faster.

By using larger values of δ, the estimate $\Delta s_{\min} = \delta^{\frac{1}{2\alpha}} S_{\min}^{\frac{1}{2\alpha^2}+\frac{1}{\alpha}}$ also provides an estimate of the linearization error outside the neighbourhood of convergence.

A. Linearization of the problem

In order to study how the algorithm works, we linearize the system by the Taylor expansion

$$x^{-\frac{1}{\alpha}} = x_0^{-\frac{1}{\alpha}} - \frac{1}{\alpha}(x - x_0)x_0^{-\frac{1}{\alpha}-1} + R_2(x)$$

around x_0, where $R_2(x) = \frac{1}{2\alpha}(\frac{1}{\alpha}+1)(x' - x_0)^2 x_0^{-\frac{1}{\alpha}-2}$ for some point x'. Suppose that $m < n$ and that $\{B_i\}_{i=1}^m$ and $\{\Gamma_j\}_{j=1}^n$ is a solution to the problem. Suppose further that β_i and γ_j are close to this solution. Hence, when defining

$$\begin{aligned} \beta_i &= B_i - x_i \text{ and} \\ \gamma_i &= \Gamma_j + y_i, \end{aligned}$$

x_i and y_j are small, and in the algorithm β_i is increasing and γ_i is decreasing. Hence, both x_i and y_j are positive and decreasing. Then we have the Taylor expansion

$$\begin{aligned} (\beta_i + \gamma_j)^{-1/\alpha} &= (B_i + \Gamma_j + y_j - x_i)^{-1/\alpha} \\ &= (B_i + \Gamma_j)^{-1/\alpha} \\ &\quad - \frac{1}{\alpha}(y_j - x_i)(B_i + \Gamma_j)^{-1/\alpha-1} \\ &\quad + O(y_j^2 + x_i^2). \end{aligned}$$

This turns the condition

$$\sum_{j \in J} w_{ij}(\beta_i + \gamma_j)^{-1/\alpha} \leq d$$

into

$$\sum_{j \in J} w_{ij}((B_i+\Gamma_j)^{-1/\alpha} - \frac{1}{\alpha}(y_j - x_i)(B_i+\Gamma_j)^{-1/\alpha-1}) - d \leq 0$$

if the remainder term R_2 is small enoúgh.

Using $\sum_{j \in J} w_{ij}(B_i + \Gamma_j)^{-1/\alpha} = d$ and the notation

$$a_{ij} = w_{ij}(B_i + \Gamma_j)^{-1/\alpha-1},$$

provides the inequalities

$$\sum_{j \in J} a_{ij}(y_j - x_i) \geq 0.$$

From the condition

$$\sum_{i \in I} w_{ij}(\beta_i + \gamma_j)^{-1/\alpha} - d \leq 0$$

we similarly obtain the inequalities

$$\sum_{i \in I} a_{ij}(y_j - x_i) \geq 0.$$

We thus obtain the system

$$\sum_{j \in J} a_{ij}y_j - x_i \sum_{j \in J} a_{ij} \geq 0 \text{ for all } 1 \leq i \leq m \quad (4)$$

$$-\sum_{i \in I} a_{ij}x_i + y_j \sum_{i \in I} a_{ij} \geq 0 \text{ for all } 1 \leq j \leq n. \quad (5)$$

This system has the obvious solution all $x_i = y_j = 0$, but the purpose of the linearization is not to solve the new problem, but to study the behaviour of the algorithm. Since we do not know the solution, we do not actually know the values of the a_{ij}:s, only that they are constant and strictly positive.

B. Error analysis

If $\sum_{j \in J} a_{ij}y_j - x_i \sum_{j \in J} a_{ij} < 0$, condition i is not feasible. We for simplicity define the quantity $E_i = -\sum_{j \in J} a_{ij}y_j + x_i \sum_{j \in J} a_{ij} > 0$ as *the non-feasibility of i*. Furthermore, $E = \sum_{i:\text{ non-feasible}} E_i$ is the *total non-feasibility*.

For the proof of Theorem 3, which ensures that the total infeasibility decreases, the following simple lemma turns out to be essential.

Lemma 2: If $m < n$, there are at most $m - 1$ γ_j's in the solution that are non-zero. Similarly, if $m > n$ there are at most $n - 1$ β_i's in the solution that are non-zero.

Proof. Assume $m < n$. We know that all machines have tight processing time for an optimal distribution, so the total processing time is md. For measuring the total processing time in terms of jobs, denote the number of tight jobs by k. Since all jobs have nonzero processing time, at least $\varepsilon > 0$, we know that

$$kd + (n - k)\varepsilon \leq md.$$

It follows from this inequality that $k \leq m - 1$. The second statement is proven similarly. ∎

Theorem 3: If the linear problem has the total non-feasibility E, and one round of the algorithm provides the new total non-feasibility E', then

$$\frac{E'}{E} \leq C < 1,$$

where C is dependent on α, w_{ij}, B_i and Γ_j only.

Proof. This proof checks the change of the non-feasibilty during one round of the algorithm. For ease of notation we define the index sets $I_j = \{i : O_{ij} \in M_i\}$ and $J_i = \{j : O_{ij} \in N_j\}$, and also

$$\begin{aligned} I'_j &= \{i : O_{ij} \in M_i : \gamma_j > 0\}, \\ J'_i &= \{j : O_{ij} \in N_j : \beta_i > 0\}. \end{aligned}$$

By Lemma 2, if $m < n$ then $I'_j \subsetneq I_j$, and if $m > n$ we know that $J'_i \subsetneq J_i$. Suppose $m > n$. Initially we suppose that all inequalities of equation 5 are satisfied, while some in 4 are not. Consider x_{i_0} to be non-feasible, and that it is adjusted into x'_{i_0} to be feasible. Then we have

$$\sum_{j \in J} a_{i_0 j}y_j - x_{i_0} \sum_{j \in J} a_{ij} = -E_{i_0} < 0.$$

and the condition is fulfilled if x_i is decreased to x'_i, where

$$x'_{i_0} = x_{i_0} - \frac{E_{i_0}}{\sum_{j \in J} a_{i_0 j}}.$$

This causes changes in the expression on the left side of 5,

$$-\sum_{i \in I} a_{ij}x_i + \frac{a_{i_0 j}E_{i_0}}{\sum_{j \in J} a_{i_0 j}} + y_j \sum_{i \in I} a_{ij},$$

which thus has non-tightness at most

$$a_{i_o j} \frac{E_{i_0}}{\sum_{j \in J} a_{i_0 j}}.$$

Then each y_j corresponding to $\gamma_j > 0$, for the set J', need to be modified into y_j' so that

$$-\sum_{i \in I} a_{ij} x_i + \frac{a_{i_0 j} E_{i_0}}{\sum_{p \in J} a_{i_o p}} - y_j' \sum_{i \in I} a_{ij} = 0,$$

where the name of a summation index has been changed from j to p to avoid confusion. Thus,

$$y_j' = y_j + \frac{a_{i_0 j} E_{i_0}}{\sum_{p \in J} a_{i_0 p} \sum_{i \in I} a_{ij}},$$

which inserted in $\sum_{j \in J} a_{ij} y_j - x_i \sum_{j \in J} a_{ij} \geq 0$ provides a new error

$$\sum_{l \in J'} a_{il} \frac{a_{i_0 l} E_{i_0}}{\sum_{p \in J} a_{i_0 p} \sum_{i \in I} a_{il}}$$

for each i. The total sum of new non-feasibilities is then

$$\sum_k E_k' = \sum_{k \in I} \sum_{l \in J'} a_{kl} \frac{a_{i_0 l} E_{i_0}}{\sum_{p \in J} a_{i_0 p} \sum_{i \in I} a_{il}}.$$

So,

$$\frac{\sum_i E_i'}{E_{i_0}} = \sum_{k \in I} \sum_{l \in J'} a_{kl} \frac{a_{i_0 l}}{\sum_{p \in J} a_{i_0 p} \sum_{i \in I} a_{il}}$$
$$= \frac{\sum_{l \in J'} a_{i_0 l}}{\sum_{j \in J} a_{i_0 j}} < 1$$

since $J' \subsetneq J$ by Lemma 2 and all $a_{ij} > 0$. If we start with more than one non-feasibilitiy, we similarly end up with a weighted sum of the individual cases, where all quantities are strictly smaller than 1. We thus have

$$\frac{\sum_i E_i'}{\sum_i E_i} \leq \max_{i_0} \frac{\sum_{l \in J'} a_{i_0 l}}{\sum_{j \in J} a_{i_0 j}} < 1$$

for the i_0 with largest infeasibility. By defining

$$C = \frac{\sum_{l \in J'} a_{i_0 l}}{\sum_{j \in J} a_{i_0 j}}$$

Theorem 3 is proved. The argument is similar in the case $m < n$. ∎

C. Proof of Theorem 1

Suppose that the roundoff error for the algorithm is δ when running the program on a specific computer. Denote $S_{ij} = \left(\frac{B_i + \Gamma_j}{\alpha - 1}\right)^{1/\alpha}$ and $\Delta s_{ij} = (s_{ij}^\alpha - S_{ij}^\alpha)^{\frac{1}{\alpha}}$. Then:

Lemma 4: The linearization is exact if all Δs_{ij}'s fulfill

$$|\Delta s_{ij}| < \delta^{\frac{1}{2\alpha}} S_{\min}^{\frac{1}{2\alpha^2} + \frac{1}{\alpha}}$$

where $S_{\min}$ is the minimal speed of the operations.

Proof. The linearization introduces no error in the algorithm if the error $R_2(y_j - x_i)$ is smaller than δ, i.e. if

$$\frac{1}{2\alpha}\left(\frac{1}{\alpha} + 1\right)(y_j - x_i)^2 (B_i + \Gamma_j)^{-1/\alpha - 2} < \delta,$$

for all i and j. Hence

$$(y_j - x_i)^2 < \delta \frac{2\alpha^2}{\alpha + 1}(B_i + \Gamma_j)^{1/\alpha + 2}.$$

By taking advantage of $S_{ij} = \left(\frac{B_i + \Gamma_j}{\alpha - 1}\right)^{1/\alpha}$ and $\Delta s_{ij} = (s_{ij}^\alpha - S_{ij}^\alpha)^{\frac{1}{\alpha}}$ we obtain $y_j - x_i = \gamma_j + \beta_i - (B_i + \Gamma_j) = \Delta s_{ij}^\alpha(\alpha - 1)$. Inserted in the estimate provides the lemma by using the fact that $1 < \left(\frac{2\alpha^2}{(\alpha+1)(\alpha-1)^2}\right)^{\frac{1}{2\alpha}} < \frac{3}{2}$ for all $2 < \alpha < 3$. ∎

Proof of Theorem 1. From Theorem 3 it follows that the number of rounds needed to reach error ε is bounded by $C \ln \frac{1}{\epsilon}$. Since the number of floating point operations at each round is bounded by a constant times mn, the error estimate in Theorem 1 follows. Finally, Lemma 4 establishes the neighbourhood statement in Theorem 1. ∎

IV. CONCLUSIONS

The polynomial time convergence of the primal-dual algorithm if $n \neq m$ is proven locally, i.e. close to the solution. The power consumption function s^α gives a convex non-linear problem, whose solution algorithm can be studied close to the unique solution by linearization. Linearization is a general method to study the complexity of an algorithm in the final phase of convergence, which of course is the most important phase. The approach also provides information that allows quantitative error estimates outside the convergence neigbourhood.

The extremely slow convergence when $n = m$, but not when for example $n = m \pm 1$, is observed in [4] and described as a result of all γ_j's β_i's being non-zero, which forces precision errors to play a significantly larger role. This can be complemented by the observation implicit in the proof of Theorem 3 that the total infeasibility at a certain step in the algorithm must not decrease at all. Since if $n = m$, Lemma 2 is not valid and neither $J' \subsetneq J$, so we cannot prove that the constant C in the proof of Theorem 3 fulfills $C < 1$.

REFERENCES

[1] Albers S., Energy-efficient algorithms. *Communications of the ACM*, 53:86-96, 2010.
[2] Albers S., Algorithms for dynamic speed scaling. In STACS'11, pages 1-11, 2011.
[3] Aydin H., Energy-aware partitioning for multiprocessor real-time systems, Proceedings of Parallel and Distributed Processing Symposium, Nice, France, 2003.
[4] Bampis E., Chau V., Letsios D., Lucarelli G., Milis I, "Energy minimization via a primal-dual algorithm for a convex program", Proceedings of 12th International Symposium on Experimental Algorithms (SEA 2013), LNCS, (Springer) (2013).
[5] Chen J., Kuo C., Energy-Efficient Scheduling for Real-Time Systems on Dynamic Voltage Scaling (DVS) Platforms, Proceedings of 13th IEEE International Conference onEmbedded and Real-Time Computing Systems and Applications 2007, Daegu, South Korea, 2007.
[6] Chen, Y., Zhang, A., Chen, G., Dong, J., Approximation algorithms for parallel open shop scheduling, *Inf. Process. Lett.* 113, No. 7, 220-224, 2013.

[7] Devanur N., Papadimitriou C., Saberi A., Vazirani V., Market equilibrium via a primal-dual algorithm for a convex program, *Journal of the ACM*, 55(5), 2008.

[8] Kuhn, H., The Hungarian method for the assignment problem, Naval Research Logistics Quarterly, 2:83–97, 1995.

[9] Lee Y. Zomaya, A.Y., Minimizing Energy Consumption for Precedence-Constrained Applications Using Dynamic Voltage Scaling, 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, 2009. CCGRID '09, Washington D.C., USA, 2009.

[10] Orlin J., Improved algorithms for computing fisher's market clearing prices: computing fisher's market clearing prices, Proceedings of the 42nd ACM symposium on Theory of computing, Cambridge, USA, 2010.

[11] Wang L., von Laszewski, G. ; Dayal, J., Wang F., Towards Energy Aware Scheduling for Precedence Constrained Parallel Tasks in a Cluster with DVFS, 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), Melbourne, Australia, 2010.

[12] Wright S., *Primal-Dual Interior Point Methods*, SIAM, 1997.