Journal of the
Military Operations Research
Society of Korea, Vol. 9, No. 1
June, 1983

A Study for Scheduling Jobs on Unrelated Parallel Processors

Kang, Sukho*
Park, Sungsoo*

Abstract

Lagrangian relaxation is used to the problem of scheduling jobs on unrelated parallel processors with the objective of minimizing makespan. The implicit condition for optimality is drawn out explicitly in order to apply the subgradient algorithm.

To obtain the optimal solution, branch-and-bound-search method is devised. In the search, the special structure of the problem is exploited effectively.

Some computational experiences with the algorithm are presented, and comparisons are made with the Land and Doig method.

1. Introduction

This paper is concerned with the scheduling of jobs on unrelated parallel processors with the objective of minimizing makespan. Parallel processor scheduling problem is to schedule $n \geq 2$ single operation jobs on $m \geq 2$ processors.

It is assumed here that jobs are initially available, non-preemptive, and precedence constraints are not existing.

Above problems can be classified according to the processor type. Identical processors imply that processing time for a job is identical on any processor. Proportional processors imply that job processing time is proportional to the processor speed. In unrelated processors, job processing time depends on particular job and processor involved.

In this paper, only unrelated processors are considered. If preemption is not allowed, the parallel processor scheduling problem is NP-complete [9]. And it is unlikely to obtain a polynomial time algorithm. Recent studies have focussed on developing performance-guaranteed heuristic algorithms.

* College of Engineering, Seoul National University

But there has been little work for unrelated processor case, and furthermore the performance of the heuristic algorithms developed is not good. The heuristics developed by Ibarra and Kim [16] have worst-case bound m, which is equal to the number of processors.

In addition to the poor performance of the heuristic algorithms, there has not been any applicable exact algorithms except two-processor case.

Though the problem can be formulated as a zero-one mixed integer program, integer programming approach has not been reported. In this paper, integer programming approach is intended. Instead of solving the problem with the existing integer programming algorithm, branch-and-bound-search algorithm is developed, using Lagrangian relaxation and subgradient algorithm.

Unrelated processor problem is more general model than that of identical or proportional processor. But the algorithm developed here cannot be applied successfully to identical or proportional processor case. Lagrangian relaxation is taken in this algorithm, but it is insufficient to obtain a good solution for identical and proportional processors. Though Lagrangian relaxation has been used extensively in a variety of problems, the model considered here is somewhat different from usual cost-minimization model. When Lagrangian relaxation is taken, there exists unconstrained variable. Therefore the subgradient algorithm cannot be used directly, because the subgradient becomes unbounded. To avoid such a situation, implicit necessary condition for optimality is stated explicitly so that the subgradient can be defined. But the solution obtained by using subgradient algorithm is not necessarily optimal. So branch-and-bound-search method is used to obtain an optimal solution. The decomposable structure of the problem is utilized, which makes the search procedure more efficient. Computer memory requirements of the algorithm are very small, since relaxed problem becomes pure zero-one integer program, search method similar to additive algorithm of Balas [2] can be used. It is necessary only to keep the data of job processing time and table of problem state.

To test the algorithm, limited number of problems were solved with the algorithm, and the same problems were solved with the Land and Doig algorithm for comparison. The results are given in section 5.

2. Related Research

For identical processor problem, LPT(Largest Processing Time) algorithm and application of Bin Packing algorithm [4] were developed as performance-guaranteed heuristic algorithms.

Sahni [20] developed exact algorithm based on dynamic programming, but no computational results were reported.

Sahni also gave ϵ-approximate algorithm based on dynamic programming and rounding technique.

Gonzalez et al. [11] and Horowitz and Sahni [15] developed heuristics for proportional processor problem. Horowitz and Sahni also developed dynamic programming algorithm, but with no computational results.

For unrelated processor problem, Horowitz and Sahni [15] gave exact dynamic programming

algorithm, Ibarra and Kim [16] developed heuristics with worst-case bound m (number of processors). Ibarra and Kim also developed heuristics for unrelated two-processor problem with worst-case bound $(\sqrt{5}+1)/2$, but they couldn't extend the idea to $m \geq 3$ processors.

For two-processor case, Bulfin and Parker [3] converted the problem into two knapsack problems, and solved it with existing knapsack algorithm.

3. Lagrangian Relaxation and Subgradient Algorithm

The integer programming formulation of minimizing makespan on identical parallel processors was introduced by Baker [1]. Slight modification of the formulation can be used for an unrelated processor case.

The formulation, called original problem, is:

$$\text{minimize} \quad y$$

$$\text{subject to} \quad y - \sum_{j=1}^{n} t_{ij}x_{ij} \geq 0 \qquad i=1,\ldots,m \qquad (1)$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad j=1,\ldots,n \qquad (2)$$

$$y \geq 0,\ x_{ij} = 0 \text{ or } 1 \text{ for all } i,j$$

The meaning of the variable is.

- m ; number of processors.
- n ; number of jobs
- y ; makespan
- t_{ij} ; processing time of job j on processor i
- x_{ij} ; 1 if job j is allocated to processor i, otherwise 0

In original problem, constraints (1) make the problem difficult to solve. So relax constraints (1) using Lagrangian relaxation. Let u be the Lagrangian vector and u_i the Lagrangian multiplier corresponding to the i-th constraint in (1).

Then, the relaxed Lagrangian becomes.

$$\text{minimize } L(u,y,x) = y - \sum_{i=1}^{m} u_i \left(y - \sum_{j=1}^{n} t_{ij}x_{ij}\right)$$

$$\text{subject to} \quad \sum_{i=1}^{m} x_{ij} = 1,\ j = 1,\ldots,n$$

$$y \geq 0,\ u \geq 0,\ x_{ij} = 0 \text{ or } 1 \text{ for all } i,j$$

In the relaxed problem, there is not variable y in the constraints. For the problem is in a minimization form, if $1 - \sum_{i=1}^{m} u_i > 0$, y is 0, if $1 - \sum_{i=1}^{m} u_i < 0$, y is unbounded.

Variable y takes a finite positive value in any feasible solution. To satisfy this requirement, the necessary condition for optimality should be $1 - \sum_{i=1}^{m} u_i = 0$. If implicit condition $1 - \sum_{i=1}^{m} u_i = 0$ is satisfied, variable y disappears from the relaxed problem. But y can be determined from the solution in the relaxed problem. Assume $\sum_{i=1}^{m} u_i = 1$ holds, take y to be anything and solve the following relaxed problem.

$$L(u) = \text{minimize } L(u,x)$$

$$= \sum_{j=1}^{n} \sum_{i=1}^{m} u_i t_{ij} x_{ij}$$

subject to $\Sigma x_{ij} = 1, j=1, ..., n$

$x_{ij} = 0$ or 1 for all i, j

The relaxed problem is decomposed into n independent subproblems, and each subproblem has a simple multiple-choice constraint.

So the solution can be easily obtained, select one variable which has a smallest objective function coefficient in the subproblem.

Let $\bar{x}$ be the solution obtained from above, and L(u) be the objective function value given u. They y can be determined from $\bar{x}$, and L(u) becomes lower bound of the original problem. Ideal Lagrangian vector u is an optimal solution of the problem

$$\text{maximize} \quad L(u)$$

$$\text{subject to} \quad \sum_{i=1}^{m} u_i = 1$$

$$u_i \geq 0, i = 1, ..., m$$

To obtain a good Lagrange multiplier vector, subgradient algorithm is used.

In the relaxed problem, the multiple choice constraints is totally unimodular. Geoffrion [10] showed that if the constraints matrix of the relaxed problem is totally unimodular, the solution value produced by Lagrangian relaxation is equal to the solution value produced by linear programming relaxation. This equality means that unless the subgradient algorithm converges to the optimal solution, the Lagrangian relaxation bound will not be as good as the linear programming relaxation bound. Since the Lagrangian relaxation bound is used as a lower bound during the branch-and-bound-search procedure, no attempt is made to achieve actual convergence. But computational experience has indicated that fairly good value could be obtained with small computational effort, which validate the use of Lagrangian relaxation and subgradient algorithm. From the solution obtained from the relaxed problem, the finish time on each processor can be determined.

Let v(i) be the finish time at processor i given fixed u, and v be the vector of v(i). Then, the relaxed problem can be written as

maximize $L(u)$

$L(u) = \min \{ u \cdot v_k ; k=1,...,K \}$

subject to $\sum_{i=1}^{m} u_i = 1,\ u_i \geq 0,\ i=1, ..., m$

Here v_k is one of the possible assignments, and K is vast number representing number of all possible assignments, and $u \cdot v_k$ denotes innerproducts. For the problem that has above form, the subgradient algorithm can be used [5].

Starting with an initial Lagrangian vector u°, the subgradient algorithm obtains u°, u^1, u^2, ... which converges asymptotically to an optimum solution.

Given u^j, the next Lagrangian vector can be obtained from

$$u^{j+1} = u^j + t_j v(u^j),\ j=0.1, ...$$

Here t_j is positive scalar, called step size, $v(u^j)$ is a subgradient vector given u^j.

t_j is given as,

$$t_j = \lambda_j \frac{L - L(u^j)}{|v(u^j)|^2},\quad \epsilon < \lambda_j \leq 2$$

The best solution obtained in a candidate problem is taken as $\overline{L}$. The candidate problem is one that obtained in the search procedure. For the value λ, a good rule is to start with $\lambda_o = 2$. If no improvement of $L(u^j)$ occurs in the last d steps, then λ is halved [8]. When the subgradient algorithm is applied, the Lagrangian vector must satisfy the condition $\sum_{i=1}^{m} u_i = 1$.

To satisfy this constraint, a projection operator P_S is used as in [14]. To say,

$$u^{j+1} = P_S (u^j + t_j v(u^j)),\qquad S \in E^n$$

Where P_S is the operator projecting E^n onto S, that is, for any $u \epsilon E^n$, the point $P_S(u)$ is the unique point of S nearest u.

The constraint $\sum_{i=1}^{m} u_i = 1$ constitute convex set, so the projection operator can be used. Given some $\bar{u}$, we must find u, minimizing $|u - \bar{u}|^2$, and satisfying $\sum_{i=1}^{m} u_i = 1$. u is the solution of the following quadratic programming problem.

$$\min \{ \frac{1}{2} |u - u|^2 ;\ \sum_{i=1}^{m} u_i = 1,\ \text{all } u_i \geq 0,\ i=1, ... m \}$$

Kuhn-Tucker necessary and sufficient condition of the above problem can be written as the following linear complementary problem.

$$u_i - \bar{u}_i = v_i - \lambda,\ v_i \geq 0,\ u_i v_i = 0 \text{ for all } i$$

Complementary pivoting algorithm can be used to solve the above problem, but the method used in [14] is adopted for the simple structure of the problem. Followings are taken from [14].

If we set $u_i(\lambda) = \max(\bar{u}_i - \lambda, 0)$, $v_i(\lambda) = \max\{-(\bar{u}_i - \lambda), 0\}$ then, the Kuhn-Tucker conditions are satisfied. So it is only necessary to find λ^* of λ for which $\sum_{i=1}^{m} u_i(\lambda) = 1$.

Suppose the coordinate of $\bar{u}$ is ordered $\bar{u}_1 \leq \bar{u}_2 \leq \ldots \leq \bar{u}_m$.

The "break-point" value of between, and outside of, which $\sum_{i=1}^{m} u_i(\lambda)$ is linear, are $\bar{u}_1, \bar{u}_2, \ldots \bar{u}_m$, and its value for $\lambda = \bar{u}_I$ $(I = 1,2, \ldots, n)$ is $\sum_{i=I+1}^{m} (\bar{u}_i - \bar{u}_I)$. Since $\sum_{i=1}^{m} \bar{u}_i(\lambda^*) = 1$ for any $\bar{u}_I < \lambda^*$, $u_I(\lambda^*) = 0$ and $\sum_{i=I+1}^{m} (\bar{u}_i - \bar{u}_I) > 1$.

Thus $u_I(\lambda^*) = 0$ for $I = 1,2, \ldots, I_0$ where

$$I_0 = \max \left\{ I ; \frac{\bar{u}_{I+1} + \ldots + \bar{u}_m - 1}{n - I} > \bar{u}_I \right\}$$

If I_0 cannot be determined from above, then I_0 is 0.

λ^* is obtained from $\sum_{i=I_0+1}^{m} \bar{u}_i(\lambda^*) = \sum_{i=I_0+1}^{m} (\bar{u}_i - \lambda^*) = 1$, i.e.

$$\lambda^* = \frac{-1 + \sum_{i=I_0+1}^{m} \bar{u}_i}{n - I_0}$$

The obtained Lagrangian vector before the projection operator is used denotes the direction where the objective value can be increased. But, the obtained point is not on the hyperplane $\sum_{i=1}^{m} u_i = 1$. The projection operator P_S takes the role to project the point on the hyperplane.

If the finish time on every processor is identical, the obtained subgradient vector is orthogonal to the hyperplane, and $u^{j+1} = u^j$ for any j. This is the case where an optimal solution is obtained. If the finish time is different, the objective value can be increased by taking relatively larger Lagrangian multiplier value to the constraint(processor) having the larger finish time. At that time, the subgradient vector is inclined to the direction where the objective value can be increased.

The projection operator projects the obtained point to the hyperplane which constitutes the necessary condition for optimality.

As the subgradient algorithm is to increase the objective value of the relaxed problem, the objective value(makespan) of the original problem decreases through the reassignment of jobs. In unrelated processor problem, the reassignment of jobs occurs rather separately according to the Lagrangian vector u.

But, in identical and proportional processor problem, all jobs are assigned to one processor in each iteration of the subgradient algorithm, which prohibits the use of Lagrangian relaxation.

At the next section, branch-and-bound-search method is used to obtain an optimal solution, but without search, the subgradient algorithm alone can be used to obtain an approximate solution and lower bound.

4. Branch-and-bound-search

The solution obtained by applying the subgradient algorithm is not necessarily optimal. There

generally exists duality gaps. So branch-and-bound-search method is used to obtain optimal solution.

This method is analogous to the search method developed by Balas [2]. Original problem is a zero-one mixed integer program, and nowadays little work has been done on applying search method to the zero-one mixed integer program. The relaxed problem becomes pure zero-one integer program, which enables the use of search method. This greatly save the computer memory requirements.

The relaxed problem is decomposed into n independent subproblem, and this characteristics can be utilized effectively in the search.

The search procedure can be depicted as a tree composed of nodes and branches. A node corresponds to a zero-one value of x, a branch joins two nodes. The two nodes differ in the state of one variable.

A variable can be in one of three states: fixed at 1, fixed at 0, or free. When a forward step is taken, a variable is fixed at 1, after a backward step a variable is fixed at 0. The search terminates when all the variables are fixed at 0. Now define the level of a node as the number of variables fixed at 1. The point x^l is the node x with l variables fixed at 1. For the relaxed problem, the level does not exceed n, because the problem is decomposed into n subproblems, and each subproblem needs only one variable to be fixed at 1.

The search method takes the following basic procedure [21].

1) Fix a free variable x_k from x^l (initially $x^l = x^\circ$) at value 1
2) Resolve the problem in the remaining free variables; then
3) Fix x_k at value 0 (or cancel x_k at level l); and
4) Repeat this process for the problem with x_k fixed at 0

The problem with some variables fixed at 0 or 1 forms a candidate problem in the branch-bound-search procedure, and the formulation is rectified according to the variable state.

For the candidate problem is formulated directly from the original problem, there is not a cumulative computational error, which can hazard the performance of the algorithm based on linear programming relaxation.

Backward step is taken when 1) there are no free variables remained 2) for the candidate problem considered, the lower bound obtained is greater than or equal to the current best solution, i.e. current incumbent 3) the candidate problem is fathomed.

When the job processing time is integer, the lower bound can be taken as $\lceil L(u) \rceil$, (smallest integer greater than or equal to L(u)), thus fathoming is occurred when $\lceil L(u) \rceil$ equals solution value of candidate problem.

In the relaxed problem, the objective function coefficient becomes rank order solution value when the corresponding variable becomes 1. Let the current incumbent be CI and lower bound of a candidate problem LB. Define $\Delta = CI - LB$, and $\delta_{ij} = \bar{t}_{ij} - \bar{t}_{kj}, i \neq k$, where t_{ij} denotes the objective function coefficient of x_{ij} in the relaxed problem, and $\bar{t}_{kj}$ denotes minimum objective function coefficient in the subproblem, i.e. best rank order solution.

If $\delta_{ij} \geq \Delta$, then x_{ij} can be fixed at 0, or cancelled at the current level. If such a variable becomes 1 at the current level, the lower bound will exceed the current incumbent. This characteristic lessens the effort to examine the unpromising nodes.

To select the branching variable at the forward step, heuristic selection criterion is used. Among the best rank order solutions in each subproblem, the one with the largest value is taken as branching variable. Such a selection criterion will increase the lower bound quickly in low level, thus many variables can be fixed at 0.

Because the relaxed problem is decomposed into independent subproblems, backward step can be taken when only the variables in one subproblem are fixed at 0. To exploit this nature, some modifications are made in the selection of branching variable.

If forward step is to occur after a backward step, the branching variable is selected in the subproblem where backward step was taken.

Thus it is necessary only to examine one subproblem at any level. If backward step was not occurred before the forward step, branching variable is selected from any subproblem by the rule stated before.

The best Lagrangian vector obtained at any level is kept for later use as an initial Lagrangian vector. If forward step is occurred, the Lagrangian vector obtained from previous level is used as an initial Lagrangian vector. If backward step is occurred, the Lagrangian vector kept previously at the same level is used. This can be a good initial Lagrangian vector because only one variable is different in state. It is necessary only to keep the problem data and table of problem state, so the memory requirements are small.

If it is sufficient to obtain practically good solution, an error tolerance can be used as fathoming criterion. If the difference of obtained solution and lower bound is within a % of lower bound, the candidate problem can be fathomed. Then, obtained solution is at least within a % of the optimal solution, and this may lessen the computational effort. Such a criterion cannot be used when lower bound is obtained by linear programming relaxation.

5. Computational Experience

The algorithm was coded in FORTRAN, and tested for limited number of problems. For comparison, branch-and-bound method developed by Land and Doig was used. The program was written by R. Shareshian in IBM Corporation. To store the tableau in the branching procedure, core memory was used.

It may be inadequate to compare the search method with the Land and Doig method which is not specifically designed for a zero-one mixed integer program. If more efficient algorithm, for example DKW algorithm [6] will be used, computation time may be reduced. But the memory requirements do not decrease. Memory requirements for the test problems were about 1 MB for the Land and Doig method, but it takes only 3 KB for the search method.

This may validate the use of the search method if limited resources are available, furthermore feasible solution is obtained at any time.

Runs were made on IBM 370-125, and the results are given in Table 1. Problem data were taken from uniform random integers between 100 and 1000.

Table 1. Computational Results for Test Problems

No. of Proc.	No. of Jobs	Problem Number	Optimal Sol. Value	Land and Doig Method		Search Method		
				CPU Time (sec)	Iteration	CPU Time (sec)	No. of Search	Approx. Sol. without Search (relative error)
2	40	1	7781	52	51	14	44	8171 (5.0%)
		2	7752	50	53	13	43	7752 (0%)
3	40	3	5205	1319	879	376	4296	5321 (2.2%)
		4	3848	750	516	238	2373	4022 (4.5%)
4	30	5	2334	1566	1664	341	2239	2474 (6.0%)
		6	2295	6935*	–	1269	9158	2453 (6.9%)
5	20	7	1229	346	640	146	664	1229 (0%)

In the table, iteration of Land and Doig method means the number of dual simplex iteration. For search method, number of search means the number of nodes where the subgradient algorithm was taken.

Though the number of nodes examined is more for the search method, computation time is about 1/3–1/5 compared to the Land and Doig method, which is not discouraging.

For problem number 6, optimal solution was obtained by the Land and Doig method, but optimality was not proven within 6935 seconds. When applying the subgradient algorithm, 150 steps were taken for the original problem, and 15 steps for the subsequent candidate problems. If the number of step is reduced, computation time for a candidate problem decreases, but the number of nodes examined may increase. When the algorithm was applied to obtain an approximate solution, i.e. only for original problem without search, obtained solution was about 5% more than the optimal solution, and optimal solution was obtained for two problems. It is worth nothing that the two-processor problems were solved easily compared to the 3 or 4-processor problems.

This fact supplements the successful experience of Bulfin and Parker [3] for the two-processor problem.

감 사

본 연구는 1982년도 문교부 해외파견 연구교수로써 수행한 연구중의 일부임을 밝히며 심심한 감사를 드립니다.

References

1. K.R. Barker, Introduction to Sequencing and Scheduling, John Wiley, 1974.
2. E. Balas, "An Additive Algorithm for Solving Linear Programs with Zero-One Variables", Operations Research, Vol. 13, pp.517-546, 1965.
3. R.L. Bulfin, R.G. Parker, "Scheduling Jobs on Two Facilities to Minimize Makespan", Management Science, Vol. 26, pp.202-214, 1980.
4. E.G. Coffman, Jr. M.R. Garey, D.S. Johnson, "An Application of Bin-Packing to Multiprocessor Scheduling", SIAM Journal on Computing, Vol. 7, 1978.
5. G. Cornuejols, M.L. Fisher, G.L. Nemhauser, "Location of Bank Accounts to Optimize Float: An Analytic Study of Exact and Approximate Algorithms", Management Science, Vol. 23, pp.789-810, 1977.
6. R.E. Davis, D.A. Kendrick, M. Weitzman, "A Branch-and-bound Algorithm for Zero-One Mized Integer Programming Problems", Operations Research, Vol. 19, pp.1036-1044, 1971.
7. M.L. Fisher, "Optimal Solution of Scheduling Problems Using Lagrange Multipliers: Part I", Operations Research, Vol. 21, pp.1114-1127, 1973.
8. M.L. Fisher, J.F. Shapiro, "Constructive Duality in Integer Programming", SIAM Journal on Applied Mathematics, Vol. 27, pp.31-52, 1974.
9. M.R. Garey, R.L. Graham, D.S. Johnson, "Performance Guarantees for Scheduling Algorithms", Operations Research, Vol. 26, pp.3-21, 1978.
10. A.M. Geoffrion, "Lagrangian Relaxation for Integer Programming", Mathematical Programming Study, Vol. 2, pp.82-114, 1974.
11. T. Gonzalez, O.H. Ibarra, S. Sahni, "Bounds for LPT Schedules on Uniform Processors", SIAM Journal on Computing, Vol. 6, No. 1 (1977).
12. M. Held, R.M. Karp, "The Traveling Salesman Problem and Minimum Spanning Trees", Operations Research, Vol. 18, pp.1138-1162, 1970.
13. M. Held, R.M. Karp, "The Traveling-Salesman Problem and Minimum Spanning Trees", Mathematical Programming 1, pp.6-25, 1971.
14. M. Held, P. Wolfe, H.P. Crowder, "Validation of Subgradient Optimization", Mathematical Programming 6, pp.62-88, 1974.
15. E. Horowitz, S. Sahni, "Exact and Approximate Algorithms for Scheduling Non-Identical Processors", Journal of the Association for Computing Machinery, Vol. 23, No. 2, 1976.
16. O.H. Ibarra, C.E. Kim, "Heuristic Algorithms for Scheduling Independent Tasks on Nonidentical Processors", Journal of the Association for Computing Machinery, Vol. 24, pp.280-289, 1977.
17. A.H. Land and A.G. Doig, "An Automatic Method of Solving Discrete Programming Problems", Econometrica, Vol. 28, pp.497-520, 1960.
18. J.A. Muckstadt, S.A. Koenig, "Scheduling in Power-Generation Systems", Operations Research, Vol. 25, pp.387-403, 1977.
19. K. Murty, Linear and Combinatorial Programming, John Wiley & Sons Inc., pp.458-466, 1976.
20. S. Sahni, "Algorithms for Scheduling Independent Tasks", Journal of the Association for Computing Machinery, Vol. 23, pp.116-127, 1976.
21. H.M. Salkin, Integer Programming,

Addison-Wesley Pub. Co., 1975.

22. D.J. Sweeney, R.A. Murphy, "A Method of Decomposition for Integer Programs", Operations Research, Vol. 27, pp.1128 1141, 1979.