[Volume 4 issue 06 June 2016]
PageNo.1469-1474 ISSN : 2320-7167

INTERNATIONAL JOURNAL OF MATHEMATICS AND COMPUTER RESEARCH

An Implementation of Parallel Computing for Hierarchical Logistic Network Design Optimization Using PSO Algorithm

Ali Amin Rashidifar[1*], Mohammad Amin Rashidifar[2], Abdolah Abertavi[3]

[1]Department of computer science, Islamic Azad University, Shadegan Branch, Shadegan, Iran
[2]Department of Mechanical Engineering, Islamic Azad University, Shadegan Branch, Shadegan, Iran
[3]Department of Electrical Engineering, Islamic Azad University, Shadegan Branch, Shadegan, Iran
*corresponding author email:shadegan_950@yahoo.com

Abstract

Recently, we have concerned the strategic optimization on logistic network design and developed an efficient two-level solution method. To cope with extremely large-scale problems, in this paper, we propose a novel algorithm for parallel computing. Thereat, noticing the analogy between the two-level algorithms and the master-slave configuration of PC cluster on one hand, and the suitability of the population-based algorithm like particle swarm optimization (PSO) on the other hand, we have developed a parallel procedure that can make overhead and idle time extremely small, and bring about high performance finally.

Keywords: Logistic network optimization, PSO, Parallel computing algorithm, Master-slave PC clusters.

1. Introduction

Noticing growing importance of supply chain management in manufacturing, we have concerned a logistic optimization that refers to mixed-integer programming problem. And we developed a two-level solution method termed hybrid Tabu search and showed its effectiveness through various applications (Wada, Shimizu & Yoo, 2005; Shimizu, Matsuda & Wada, 2006; Shimizu, Wada & Yamazaki, 2007).

To cope with extremely large-scale and complicated problems encountered in real-world applications, this study proposes a novel algorithm for the parallel computing aiming at time reduction and improvement of solution quality at the same time. For this purpose, we employed particle swarm optimization (PSO) for solving the upper level sub-problem instead of Tabu search used in the previous studies. Population-based algorithm of the PSO seems particularly suitable for the parallel computing in accordance with the present goal and circumstance. After showing a modified algorithm of PSO to deal with the binary variables standing for open or close of the sites, we will outline the algorithm and discuss its properties.

Finally, we provide a few numerical experiments to validate the effectiveness of the proposed method.

2. Parallel Computing for Logistic Optimization

2.1. Strategic Logistics Network Model

Let us take a logistic network composed of plant, distribution center (DC), and customer as shown in Figure1. Then consider an optimization problem formulated as the following mixed-integer programming problem.

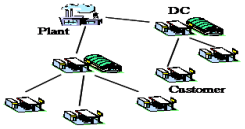

Figure1. Logistic networks concerned here

$$Min \quad \sum_{i\in I}\sum_{j\in J}(T1_{ij}+C_i)\cdot e_{ij}+\sum_{j\in J}\sum_{k\in K}(T2_{jk}+H_j)\cdot f_{jk}+\sum_{j\in J}F_j\cdot x_j$$

$$\text{Subject to} \begin{cases} \sum_{j \in J} f_{jk} = D_k \quad \forall k \in K & (1) \\ \sum_{i \in I} e_{ij} = \sum_{k \in K} f_{jk} \quad \forall i \in I & (2) \\ \sum_{j \in J} e_{ij} \le S_i \quad \forall i \in I & (3) \\ \sum_{k \in K} f_{jk} \le U_j \cdot x_j \quad \forall j \in J & (4) \\ x_j \in \{0,1\} \quad \forall j \in J \\ e_{ij}, f_{jk} \in \{\text{Non negative real number}\}, \ \forall i \in I, \ \forall j \in J, \ \forall k \in K \end{cases}$$

Where notations denote as follows:

C_i: production cost per unit amount at plant i
D_k: demand of customer k
e_{ij}: shipped amount from plant i to DC j
F_j: fixed-charge cost for opening DC j
f_{jk}: shipped amount from DC j to customer k
H_j: holding cost per unit amount at DC j
S_i: upper bound for production at plant i
$T1_{ij}$: transport cost from plant i to DC j per unit amount
$T2_{jk}$: transport cost from DC j to customer k per unit amount.
U_j: upper bound of holding capacity at DC j
x_j: take 1 if DC j is open, otherwise 0
I, J, K: index set of plants, DCs and customers, respectively

The objective function is the total cost composed of transportation costs, production costs at plant, operational costs at DC, and fixed-charge costs for opening the DC. On the other hand, we impose the constraints on demand satisfaction of every customer Eq.(1), input-output balance at each DC Eq.(2), available amount of product from each plant Eq.(3), and holding capacity bound at each DC Eq.(4). In addition, binary decision variables are introduced for the selection of open DC while non-negative real variables for transport amount.

To solve this kind of problem, we applied successfully the hybrid Tabu search. Its development relies on the fact that metaheuristic method is amenable for solving the upper-level sub-problem that refers to a combinatorial optimization. On the other hand, the lower-level problem reduces favorably to the linear program after the binary variables are fixed. Additionally, it can be further transformed into the minimum cost flow problem that is solvable by the graph algorithm much faster than any other mathematical programming method.

Furthermore, noticing the analogy between such two-level algorithms and the master-slave configuration of the PC cluster, we can realize an appropriate framework for the parallel computing. Thereat, the master PC engages in the decision of DC locations that corresponds to the upper-level sub-problem while each slave tries to optimize the route selection in the lower-level sub-problem under the DC sites allocated by the master.

Moreover, we employ the PSO for solving the location sub-problem after giving a modified algorithm mentioned below to deal with the binary decision variables. Its population-based algorithm can help reach the global optimum more efficiently than a local search-based algorithm like the Tabu search.

2.2. Modified PSO for 0-1 Programming Problem

The PSO is a metaheuristic optimization technique developed recently (Kennedy and Eberhart, 1995, 1997) after the behavior of bird flocking or fish schooling (swarm), and known as a powerful global optimization method with respect to real valued-variables.

Members of swarm communicate with each other and adjust their own positions and velocities based on the information regarding the good positions both of their own and swarm.
In practice, the position and the velocity are updated through the following formulas, respectively.

$$x_i(t+1) = x_i(t) + v_i(t+1), \tag{5}$$

$$v_i(t+1) = w \cdot v_i(t) + r_1 b(p_i - x_i(t)) + r_2 c(y_n - x_i(t)) \quad (i = 1,2,\ldots,N_p), \tag{6}$$

Where t is generation, N_p a swarm size, w an inertial constant, b and c are constants emphasizing how to guide each member to a good position. Moreover, r_1 and r_2 are random values in the range [0, 1], p_i is the best position seen by the boid i (personal best), and y_n the global best position seen by the swarm (net best).
The algorithm is simple enough as outlined below.

Step 1: Set t=1. Initialize $x_i(t)$ and $v_i(t)$ randomly within the admissible range of these values, each p_i to the current position, and y_n to the position having the best fitness among the swarm.
Step 2: For each member, do the following: obtain $x_i(t+1)$ and $v_i(t+1)$ according to Equation (5) and (6), respectively, and evaluate the new position. If it outperforms p_i, update p_i, and if it outperforms y_n, update y_n.
Step 3: If the stopping condition is satisfied, stop. Otherwise let $t=t+1$, and go back to Step 2.

Now, to cope with the 0-1 programming problem, we modified the original method (discrete PSO). First, let us give a binary code of decision vector describing whether the site is open (1) or close (0). Figure 2 exemplifies such coding for y_n, p_i and $v_i(t)$, respectively. Then viewing that the calculation by Equations (5) and (6) is a sort of real-value relaxation of Boolean algebra, we attempt to update the velocity and position as follows:

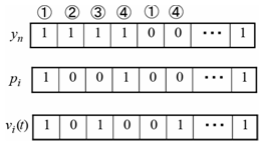

Figure2. Coding and four different bit patterns

In bit-wise, apply the majority rule among the elements $v_i(t)$, p_i, y_n, and decide $v_i(t+1)$ based on the vote with the different probabilities r_a, r_b, r_c and r_d depending on the occurrence among them (There occur four different cases as shown in Fig.2.).
Here r_a is a probability when $v_i(t)=p_i=y_n$ (Case 1), r_b when $v_i(t)=p_i\neq y_n$ (Case 2), r_c when $v_i(t)=y_n\neq p_i$ (Case 3), r_d when $v_i(t)\neq(y_n=p_i)$ (Case 4). In tuning of these parameters, therefore, the following relations should be held: $1>r_a>r_b>r_c>r_d>0$.
Likewise, if $x_i(t)=v_i(t+1)$, then let $x_i(t+1)$ be $x_i(t)$ with probability r_e. Otherwise let it be $v_i(t+1)$ with r_f. Here set the probability rate so that $1>r_e>r_f>0$.
Moreover, magnitude of the velocity is considered by the number of locus where the bit operation mentioned above is undertaken. It should be decreased linearly or non-linearly along with the generation.

2.3. Algorithm for Parallel Computing by Discrete PSO

In what follows, we outline the tasks assigned to master and slave PCs, respectively.

Master PC

Step 1: Notify the number of DC location that is different per each slave PC.
Step 2: Enter "at call" mode
Step 3: If contact from the slave is a request for notification of the latest net best, inform it to the slave. Then, go to Step 4. If the contact is the report of new finding, compare it with the current net best. Then update the current best if it outperforms the current one. Otherwise return the current net best to the slave.

Step 4: If a certain convergence condition is satisfied, let every slave stop, and the net best be the final solution. Otherwise go back to Step 2.

Slave PC

Step 1: According to the notified DC number, allocate the initial location.

Step 2: Solve the route selection problem. If the current search finds the personal best, update it. In addition, if it outperforms the net best on hand, report it to the master PC. Otherwise, contract with the master without doing anything.

Step 3: Ask the master to inform the latest net best.

Step 4: Re-locate the DC sites based on the PSO algorithm mentioned already and go back to Step 2.

2.4. Evaluation of the Parallel Computing

There are popularly known several factors that will affect on the performance of the parallel computing, i.e., loads imbalance between master and slave, granularity and frequency of information exchange, and overhead for the parallelism. Regarding these factors, our algorithm has nice properties. First, it can provide a wonderful timing chart that enables us to avoid almost completely the idle time due to the load imbalance. Regarding the communication between the master and slave, not only its amount is small (only a binary code standing for DC location and the objective value) but also frequency is low due to the multi-walk implementation (The search starting from different point is leaved totally to each slave PC). Moreover, synchronization regarding information exchange between every PC is unnecessary as known from Fig.3. That can produce an additional effect on the performance of metaheuristic. Since each member is to be controlled by the different net best (See Eq.(6)), by the virtue of asynchronization, we can increase the manifoldness essential for the metaheuristic algorithm without paying any particular attentions. Due to these effects, we can make the overhead for the parallelism very small, and realize the high performance algorithm.

To evaluate the total performance of the parallel computing, the following two indices are commonly used. When it takes $T(P)$ CPU time using P number of PCs, speedup rate $S(P)$ is defined by $S(P) = T(1)/T(P)$, and the efficiency η by $\eta = S(P)/P$. Hence, ideally these values become $S(P)=P$ and $\eta = 1$, respectively.

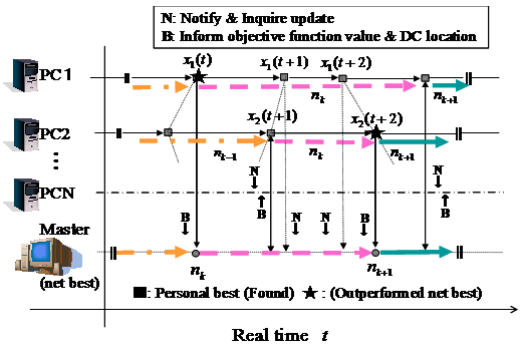

Figure3. Timing chart of communication between master and slave PCs

3. Numerical Experiments

Using up to 9 PCs, one of which works as the master while the others as slaves, we performed the numerical experiments under the environment on a cluster of Athlon 500 MHz processor and 100 Mbps network speed, running under Debian Linux and using the library of MPICH (Pacheco, 1997).

Table 1 Comparison with the conventional method

Parameter size[1] (Problem size)[2]	ID	Objective value		CPU time		Stopping generation
		PSO	M-Tabu	PSO	M-Tabu	

30, 300, 1,000		A1	33492.5	33386.0	1892.8	2426.4	16,000
(300, 1,630)	309,000,	A2	28707.1	28594.7	1758.6	2290.9	
40, 400, 1,200		B1	36473.1	36553.8	3408.1	4312.2	16,000
(400, 2,040)	496,000,	B2	30649.2	30568.9	2889.1	3952.2	
50, 500, 1,500		C1	39369.5	39342.5	7388.4	9346.0	24,000
(500, 2,550)	775,000,	C2	36840.3	36760.8	6954.0	8702.9	

[1] Number of plants, DCs and customers, [2] Number of binary and real variables, and constraints

We prepared totally six benchmark problems with 3 different problem sizes two each. Table 1 summarizes the results (averaged over five trials) in comparison with another method that was developed based on the multi-start Tabu search with information sharing and exchanges (M-Tabu in Table 1 & Fig.4). Though the M-Tabu method could outperform the conventional sequential algorithm in some numerical experiments, it was implemented sophisticatedly based on the minute preliminary investigations (Ohbora, 2005). Hence, another advantage of the proposed approach is said to be its simplicity and plainness tuning the algorithm compared with the M-Tabu method. From Table 1, we know the proposed method achieves nearly the same Performance with less computation time. In addition, convergence property is known better than the conventional result as shown in Fig.4 (stopping condition is the prescribed generation).

Finally, the speedup rate and the efficiency in Table 2 also validate the effectiveness of the proposed method. Thereupon the parallelism effect along the increase in the number of PC is known to be splendid nevertheless the effect is discounted by the idle master PC. To increase the efficiency in the further development, just an extra shared memory or blackboard is enough in instead of the master PC.

From these facts, we can expect to utilize the proposed method to solve much larger scale problem efficiently using larger cluster.

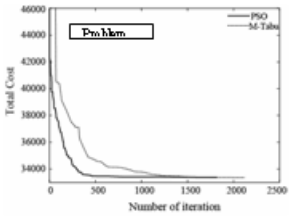

Figure4. Convergence trend along generation

Table 2 Numerical results of the parallelism performance

P	Obj. value	CPU time	Speedup *S*(*P*)	Efficiency η
1	33433.5	15209.9	-	-
2+master PC	33519.3	7448.2	2.04	0.68
4+master PC	33526.2	3658.4	4.16	0.83
8+master PC	33492.5	1892.8	8.04	0.89

4. Conclusions

To solve the strategic optimization problem for large scale logistic network design, we have developed a practical and efficient method for the parallel computing and implemented it as a master-slave configuration.

Thereupon, we are interested in the PSO and applied it after giving its modified algorithm to handle the binary variables.
In our knowledge, this is the first and novel approach of PSO for realizing information sharing and exchange and multi-walk in parallel computing. Additionally, due to the analogy between the two-level solution algorithms and the master-slave configuration of PC cluster, the developed procedure can provide very nice properties regarding the reduction of the overhead and the idle time in the parallel computing.
Through numerical experiments, effectiveness of the proposed procedure is confirmed. Relying on these results, the proposed approach is shown promising for large-scale and complicated real world applications targeting at global logistics of chemical industries.

References

[1] J. Kennedy and R. C. Eberhart, 1995, Particle Swarm Optimization, Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ.pp. 1942-1948
[2] J. Kennedy and R. C. Eberhart, 1997, A Discrete Binary Version of the Particle Swarm Algorithm, Proceedings of World Multiconference on Systemics,Cybernetics and Information, Piscataway, NJ.pp. 4104-4109
[3] T. Ohbora, 2005, Bacheor Thesis of Toyohashi University of Technology.
[4] P. S. Pacheco, 1997, Parallel Programming with MPI, Morgan Kaufmann Publisher.
[5]Y. Shimizu, S. Matsuda and T. Wada, 2006, A Flexible Design of Logistic Network against Uncertain Demands through Hybrid Meta-Heuristic Method, Proc. 16th Europe. Symp. On Computer-Aided Process Eng., Garmisch Partenkirchen, Germany, pp. 2051-2056
[6] Y. Shimizu, T. Wada, Y. Yamazaki, 2007, Logistics Optimization Using Hybrid Meta heuristic Approach under Very Realistic Conditions, Proc. 17th Europe. Symp. on Computer-Aided Process Eng., Bucharest, Romania
[7] T. Wada, Y. Shimizu and J.K. Yoo, 2005, Entire Supply Chain Optimization in Terms of Hybrid in Approach, Proc. 15th Europe. Symp. on Computer-Aided Process Eng., Barcelona, Spain, pp. 1591-1596