Journal of Computer Science 5 (1): 64-70, 2009
ISSN 1549-3636

Benders' Decomposition Based Heuristics for Large-Scale Dynamic Quadratic Assignment Problems

[1]Sirirat Muenvanichakul and [2]Peerayuth Charnsethikul
[1]Faculty of Engineering Si Racha, Kasetsart University Si Racha Campus, Chonburi 20230 Thailand
[2]Department of Industrial Engineering, Kasetsart University, Bangkok, 10903 Thailand

Abstract: Problem statement: Dynamic Quadratic Assignment Problem (DQAP) is NP hard problem. Benders decomposition based heuristics method is applied to the equivalent mixed-integer linear programming problem of the original DQAP. **Approach:** Approximate Benders Decomposition (ABD) generates the ensemble of a subset of feasible layout for Approximate Dynamic Programming (ADP) to determine the sub-optimal optimal solution. A Trust-Region Constraint (TRC) for the master problem in ABD and a Successive Adaptation Procedure (SAP) were implemented to accelerate the convergence rate of the method. **Results:** The sub-optimal solutions of large-scales DQAPs from the method and its variants were compared well with other metaheuristic methods. **Conclusion:** Overall performance of the method is comparable to other metaheuristic methods for large-scale DQAPs.

Key words: Dynamic quadratic assignment, benders decomposition, dynamic programming, trust region method

INTRODUCTION

A Dynamic Quadratic Assignment Problem (DQAP) is a decision problem of finding the optimal location assignments among a set of facilities over a set of discrete periods. The objective is to minimize the sum of flow costs and rearrangement cost over all discrete time periods.

DQAP can be mathematically formulated as the following modified QAP:

Minimize:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n}\sum_{t=1}^{T} C_{ijklt}X_{ijt}X_{klt} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n}\sum_{t=1}^{T-1} R_{ijlt}X_{ijt}X_{il(t+1)} \quad (1)$$

Subject to:

$$\sum_{i=1}^{n} X_{ijt} = 1, \forall j, t \quad (2)$$

$$\sum_{j=1}^{n} X_{ijt} = 1, \forall i, t \quad (3)$$

$$X_{ijt} \in \{0,1\}, \forall i, j, t \quad (4)$$

Where:

- n = The number of facilities/locations in each period t
- T = The number of discrete time periods
- $C_{ijklt} = f_{ikt} * d_{jlt}$ = The cost of assigning facility i to location j and facility k to location l at period t
- f_{ikt} = The workflow cost from facility i to facility k at period t
- d_{jlt} = The distance from location j to location l at period t
- R_{ijlt} = The rearranging cost when facility i located on location j at period t is moved to location l at period (t+1)
- X_{ijt} = 1, if facility i is assigned to location l at period t. Otherwise, $X_{ijt} = 0$

This study presents an alternative approach to large-scale DQAPs. To take a full advantage of existing, well-developed exact algorithm, DQAP is transformed into an equivalent linear problem. Our approach is based on Benders Decomposition (BD) and Dynamic Programming (DP). Relaxations are introduced in order to make the methods competitive for large-scale problems. Different accelerating techniques including trust-region and a Successive Adaptation Procedure (SAP) are examined.

Corresponding Author: Sirirat Muenvanichakul, Faculty of Engineering at Si Racha, Kasetsart University Si Racha Campus, Chonburi 20230 Thailand

MATERIALS AND METHODS

An equivalent linear problem: The linearization of QAP can be modified for the DQAP by introducing two new variables. The variable Y_{ijklt} is equal to one if at period t, the facility i is assigned to site j and facility k is assigned to site l. Another variable is $V_{ijl(t+1)}$ representing the facility i located to site j at period t and relocated to site l at period t+1. Therefore, the linear transformed problem of (1-4) becomes:

Minimize:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n}\sum_{t=1}^{T} C_{ijklt} Y_{ijklt} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n}\sum_{t=1}^{T-1} R_{ijlt} V_{ijl(t+1)} \quad (5)$$

Subject to:

$$Y_{ijklt} \geq X_{ijt} + X_{klt} - 1,\ i = 1,..,n,\ j = 1,\ldots,n,\ t = 1,\ldots T \quad (6)$$

$$V_{ijl(t+1)} \geq X_{ijt} + X_{il(t+1)} - 1, i = 1,..,n,\ j = 1,\ldots,n,\ t = 1,\ldots T \quad (7)$$

$$\sum_{i=1}^{n} X_{ijt} = 1,\ j = 1,\ldots,\ n,\ t = 1,\ldots T \quad (8)$$

$$\sum_{j=1}^{n} X_{ijt} = 1,\ i = 1,\ldots,\ n,\ t = 1,\ldots T \quad (9)$$

$$Y_{ijklt}, V_{ijl(t+1)} \geq 0\ ,\forall\ i, j, t \quad (10)$$

$$X_{ijt} \in \{0,1\},\forall\ i, j, t \quad (11)$$

Let the DQAP defined in (1-4) be designated problem Q and the Mixed-Integer Linear Programming (MILP) defined in (5-11) be designated problem L. By extending the theorem and proof in Lawler[1], the equivalence of Q and L for any given set of cost coefficients can be assured.

Benders Decomposition (BD): Using the BD algorithm, presented in Benders[2], the linearized DQAP (5-11) can be decomposed into:

A linear programming sub-problem (dual problem):
Maximize:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n}\sum_{t=1}^{T}(X^{*}_{ijt} + X^{*}_{klt} - 1)U_{ijklt} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n}\sum_{t=1}^{T-1}(X^{*}_{ijt} + X^{*}_{il(t+1)} - 1)V_{ijl(t+1)} \quad (12)$$

Subject to:

$$0 \leq U_{ijklt} \leq C_{ijklt},\ i = 1,..,n,\ j = 1,\ldots,n,\ t = 1,\ldots T \quad (13)$$

$$0 \leq V_{ijl(t+1)} \leq R_{ijlt},\ i = 1,..,n,\ j = 1,\ldots,n,\ t = 1,\ldots T \quad (14)$$

for a given layout $X^{*}_{ijt} \in \{0,1\}\ ,\forall\ i, j, t,$ and:

A mixed-integer-linear-programming master-problem:
Minimize:

$$Z \quad (15)$$

Subject to:

$$Z \geq \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n}\sum_{t=1}^{T}(X_{ijt} + X_{klt} - 1)U^{*}_{ijklt} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n}\sum_{t=1}^{T-1}(X_{ijt} + X_{il(t+1)} - 1)V^{*}_{ijl(t+1)} \quad (16)$$

$$\sum_{i=1}^{n} X_{ijt} = 1,\ j = 1,\ldots,\ n\ ,\ t = 1,\ldots T \quad (17)$$

$$\sum_{j=1}^{n} X_{ijt} = 1,\ i = 1,\ldots,\ n\ ,\ t = 1,\ldots T \quad (18)$$

$$X_{ijt} \in \{0,1\},\forall\ i, j, t, \quad (19)$$

for given all possible values of U^{*}_{ijklt} and $V^{*}_{ijl(t+1)}$.

The upper bound and the lower bound of the problem are defined respectively as:

$$UB = \min(UB_{current}, \text{Objective value of the sub-problem})$$

and

$$LB = \max(LB_{current}, \text{Objective value of the master-problem})$$

In exact algorithm, BD solves the sub-problem and master problem iteratively until:

$$UB - LB \leq \delta_{BD} \quad (20)$$

where, δ_{BD} is a given tolerance.

At the end of each iteration, a new cut is added to constraint (16) in the master problem.

Since constraints (13 and 14) are always feasible, there is no feasibility cut in BD of the linearized DQAP

(5-11). Furthermore, this implies that the solution of the sub-problem can be determined directly from:

$$U^{*}_{ijklt} = \begin{cases} C_{ijklt} \text{ if } \left(X_{ijt} + X_{klt} - 1\right) = 1 \\ 0 \qquad \text{otherwise} \end{cases} \tag{21}$$

And

$$V^{*}_{ijl(t+1)} = \begin{cases} R_{ijlt} \text{ if } \left(X_{ijt} + X_{il(t+1)} - 1\right) = 1 \\ 0 \qquad \text{otherwise} \end{cases} \tag{22}$$

Approximate Benders Decomposition (ABD): An aggressive relaxation to the exact BD algorithm is taken. The solution of master problem is approximated by a linear programming whose solution is a real number from 0-1. Hungarian algorithm is utilized to round the solution to 0 or 1. The cost function in Hungarian algorithm is defined as the difference between the solution from the linear programming and 1. Note that this ABD does no longer guarantee the optimal solution after adding all the cuts as the exact BD does.

Furthermore the maximum number of allowable cuts Ncut_max is added as the other stopping criterion for large-scale problems. ABD is likely to stop because the number of iterations has reached the maximum number of allowable cuts.

Trust-Region Constraint (TRC): An additional constraint for trust-region to regulate the change of the solutions in the two consecutive iterations in the master problem is introduced into the master problem of the ABD. As proposed by Santoso[3], the additional constraint imposed at $p+1^{th}$ iteration bounds the Hamming distance of the master problem solution from the solution at p^{th} iteration:

$$\sum_{q \in X^{p}} \left(1 - x_{q}^{p+1}\right) + \sum_{q \notin X^{p}} x_{q}^{p+1} \le \Delta^{p+1} \tag{23}$$

Where:

Δ^{p+1} = The trust-region size at $p+1^{th}$ iteration

X^{p} = The master problem solution obtained at p^{th} iteration and let $X^{p} = \{q : x_{q}^{p} = X_{ijt}^{p} = 1\}$

The trust-region size is kept constant through out each solution procedure in this study.

Approximate Dynamic Programming (ADP): With Rosenblatt's dynamic programming model[4], each period corresponds to a stage and each layout arrangement corresponds to a state. Therefore, there are n! states in each of the T stages. The total number of possible solutions is $(n!)^{T}$. The bounding procedure reduces the number of candidate static layouts to be examined by including only best static layouts for each period defined by any layout arrangement (for a given period) that has the difference between the total cost of the arrangement and the cost of optimal static solution for that period lower than the difference between the values of the upper bound and the lower bond of the model.

In large-scale problems, it is impossible to include all $(n!)^{T}$ possible layouts in the bounding procedure. ADP relies on the ensemble of sample static layouts given to the bounding procedure. There is no guarantee that the solution from ADP is the optimal solution. The quality of the sub-optimal solution from ADP depends strongly on the quality of the ensemble of sample static layouts.

Combinatorial Method and Successive Adaptation Procedure (SAP): The combinatorial method makes use of ABD to generate the ensemble of sample static layouts given to ADP. ADP performs the bounding procedure and searches for the sub-optimal solution. Iterations over ABD and ADP loop are introduced to further improve the quality of the sub-optimal solution. The iterations continue until:

$$\frac{\text{Total Cost}_{\text{current}} - \text{Total Cost}_{\text{previous}}}{\text{Total Cost}_{\text{previous}}} \times 100 \le \delta_{\text{opt}} \tag{24}$$

where, δ_{opt} is a given tolerance. From our unpublished study, the final sub-optimal solution is not sensitive to the exact value of δ_{opt} in (24); δ_{opt} is set to be to 0.001% though out our study.

Another way to improve the quality of the solution is to include as many static layouts in the ensemble as possible. In this study, the ensemble is expanded by including unique static layouts from different time periods obtained from ABD layouts and appending the new set of unique static layouts to the ensemble in the previous iteration.

The combinatorial method is further enhanced by a SAP. In SAP, ABD and ADP loops are implemented successively with different trust-region-sizes and the numbers of maximum allowable cuts in ABD. For a given initial layout, the procedure starts with a small number of maximum allowable cuts and large trust-region size. The sub-optimal solution from the previous iteration is used as an initial layout for the method with

a smaller trust-region size to regulate the sampling process in ABD to be in the neighbor of a good initial layout from the previous step. The procedure continues successively until no further improvement is achievable or termination by the user. The procedure may continue successively with a larger number of maximum allowable cuts with a successive trust-region size reduction procedure.

Implementation: All algorithms are implemented in MATLAB version R2007b and all the experiments are done on a notebook with 1.66 GHz. Intel®Core 2 CPU, 1 GB RAM with Window XP 64 bits. All test data are available upon request.

RESULTS

Combinatorial Method (ABD+ADP): The total cost of the DQAP of the size n = 20, T = 5 from the combinatorial method without the trust-region constraint or the adaptive procedure (ABD+ADP) with the number of maximum allowable cuts Ncut_max of 500 is shown in Table 1 against the total costs from other metaheuristic methods including Simulated Annealing (SA)[5], Genetic Algorithm (GA) and Tabu Search (TS)[6]. The initial layout for ABD+ADP is arbitrary and set to be $X_{iit} = 1, \forall i, t$. The possibility of improving the quality of the sub-optimal solution from ABD+ADP by using a better initial layout from SA is investigated. The total cost is also shown in Table 1.

Figure 1 shows the best total cost from ABD (the upper bound of ABD) and the total costs from ADP from all iterations before the solution from ABD+ADP converges to its sub-optimal value. The initial layout in this case is set to be $X_{iit} = 1, \forall i, t$ as well.

Table 1: Cost (units) and CPU time (seconds) for DQAP by ABD+ADP

	Problem size	
Algorithms	n = 20, T = 5	n = 40, T = 3
ABD+ADP (initial layout -arbitrary; ncutmax = 500)	4,416,613 (366,872 sec)	11,203,203 (not converged)
ABD+ADP (initial layout-SA; ncutmax = 500)	4,383,877 (39,845 sec)	10,872,483 (6,773 sec)
TRC (initial layout – SA; Ncutmax = 20; TR = 0.5)	4,360,788 (712 sec)	10,834,881 (892 sec)
SAP (initial layout-arbitrary)	4,370,302 (18,239 sec)	11,015,498 (43,910 sec)
SAP (initial layout-SA)	4,316,387 (11,459 sec)	10,817,002 (16,359 sec)
SA	4,383,877	10,872,483
TS (Initial Layout-Random)	4,399,513	11,245,315
TS (Initial Layout-SA)	4,256,480	10,761,358
GA (Initial Layout-Random)	4,605,719	11,358,612
GA (Initial Layout-SA)	4,361,559	10,854,512

The effect of the number of maximum allowable cuts Ncut_max on the performance of ABD+ADP is shown in Table 2. The initial layout in this case is set to be $X_{iit} = 1, \forall i, t$. It is clear from the Table that the executable time for ABD+ADP with Ncut_max of 500 is unreasonably long (more than 4 days) making it is impractical for large-scale problems. From this point of view, ABD+ADP with Ncut_max of 20 is adopted for further development.

ABD+ADP is further tested with the DQAP of the size n = 40, T = 3. The calculation has not fully converged for an arbitrary layout $X_{iit} = 1, \forall i, t$. It terminated due to memory overflowing. Nevertheless, the unconverged result (Table 1) is comparable with other metaheuristic methods and exhibit similar feature as the problem of the size n = 20, T = 5. The total executable time is not available for this set of experiment.

Trust-Region Constraint (TRC): TRC is tested with n = 20,T = 5. The maximum number of allowable cuts is set to 20. The trust-region size, Δ^{p+1}, is set to be 0.5 corresponding to allowing only 25% of the layout to be changed. The sub-optimal layout from SA is used as an initial layout. The result and the executable time are also shown in Table 1.

Table 2: Cost (units), CPU time (seconds) and the number of sample layouts in ADP for DQAP with different number of maximum allowable cuts in ABD (DQAP: n = 20, T = 5; initial layout: Arbitrary)

Algorithm	Ncut_max	Cost (units)	CPU (sec.)	#L/O
ABD+ADP	10	4,519,097	118.74	337
	20	4,485,903	171.02	463
	40	4,494,882	640.11	601
	500	4,416,613	366,872.16	14,954
	500(SA)	4,256,480	47,552.52	2,505

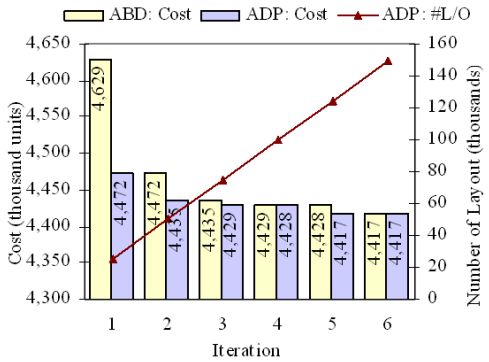

Fig. 1: ABD and ADP Costs (units) from ABD + ADP and the number of sample layouts (#L/O) in ADP. (DQAP: n = 20, T = 5; initial layout: Arbitrary; Ncut_max = 500)

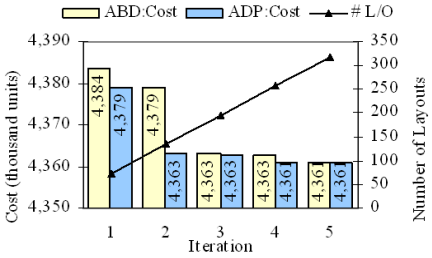

Fig. 2: ABD and ADP costs (units) from TRC and the number of sample layouts in ADP. (DQAP: n = 20, T = 5; initial layout: SA; Ncut_max = 20, TR_size_constant = 0.5)

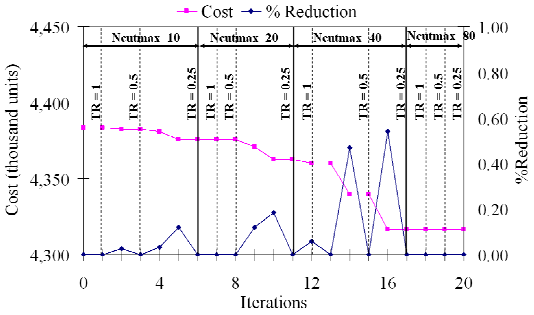

Fig. 3: Cost (units) and cost reduction (%) over each iteration of SAP. (DQAP: n = 20, T = 5; initial layout: SA)

The same test with the same parameters is carried out with the problem of the size n = 40, T = 3. The result is also shown in Table 1.

Successive Adaptation Procedure (SAP): TRC is implemented successively by increasing the number of maximum allowable cuts Ncut_max in the following order: 10, 20, 40 and 80 and reducing the trust region size Δ^{p+1} TR_size_constant in (24) in the following order: 1, 0.5 and 0.25. The total cost of SAP starting from the sub-optimal layout from SA with n = 20, T = 5 is shown in Table 1. The total cost and cost reduction during SAP is shown in Fig. 3. SAP takes 11,459 seconds (over 3 h) to complete the procedure shown in Table 1. Figure 4 shows the best total cost from ABD (the upper bound of ABD) and the total costs from ADP during SAP.

The total cost and cost reduction during SAP for arbitrary initial layout ($X_{iit} = 1, \forall i, t$) is shown in Table 1 and Fig. 5. The best total cost from ABD (the upper bound of ABD) and the total costs from ADP during SAP for this case shown in Fig. 6, quite similar to other methods above. The executable time (Table 1) for this case is 18,239 sec (more than 5 h).

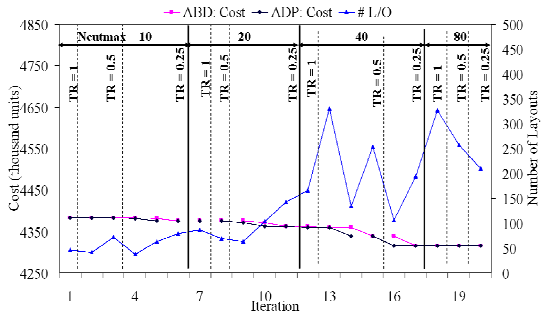

Fig. 4: ABD and ADP Costs (units) and Cost Reduction (%) over each iteration of SAP. (DQAP: n = 20, T = 5; initial layout: SA)

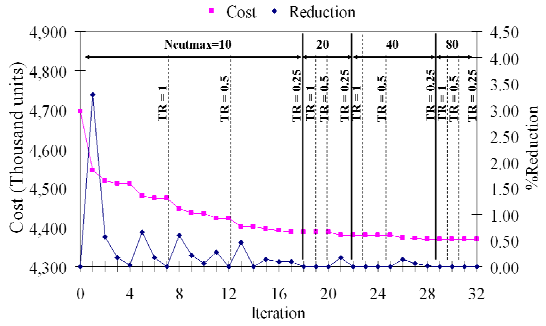

Fig. 5: Cost (units) and cost reduction (%) over each iteration of SAP. (DQAP: n = 20, T = 5; initial layout: Arbitrary)

Table 3: The result summary provided by ABD+DAP with SAP

Problem size		Cost of Int. L/O	Total cost		
n	T	(units)	Best cost (units)	Reduction (%)	Exe time (sec)
20	3	2,904,999*	2,724,991	6.1965	8,335.3
20	5	4,694,840*	4,370,302	6.9127	18,238.8
20	8	7,491,424*	6,887,656	8.0595	122,621.4
40	3	11,601,354*	11,015,498	5.0499	43,910.1
40	5	19,247,187*	18,131,694	5.7956	103,756.6
20	5	4,383,729**	4,316,387	1.5362	11,459.1
40	3	10,862,694**	10,817,002	0.4206	16,359.3

SAP has also applied to DQAP with different sizes. Because there is no available sub-optimal solution from other methods, SAP starts from an arbitrary initial layout $X_{iit} = 1, \forall i, t$. The initial cost, sub-optimal total cost, over cost reduction and executable time are shown in Table 3 (SAP starting with a sub-optimal layout from SA for n = 20, T = 5 and n = 40, T = 3 problems are also included for the sake of reference). The total cost reductions are in the neighbor of 5-8%. The executable time varies from 8,335 sec (slightly more than 2 h) for n = 20, T = 3 problem to 122,621 sec. (more than 34 h) for n = 20, T = 8 problem.

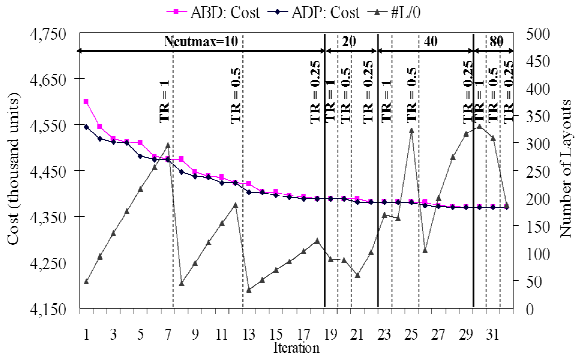

Fig. 6: ABD and ADP Costs (units) and Cost Reduction (%) over each iteration of SAP. (DQAP: n = 20, T = 5; initial layout: Arbitrary)

DISCUSSION

ABD+ADP: The total cost from ABD+ADP is 4,416,613 (Table 1) compared well to that from other metaheuristic methods starting from random initial layouts. In fact, ABD+ADP produces a lower total cost than the total cost from GA with a random initial layout. However, TS and GA slightly outperform ABD+ADP, if the best layout from SA is supplied in as initial layouts for the two methods. The result of ABD+ADP starting from SA sub-optimal layout shown in Table 1 suggesting that ABD+ADP fails to take advantage of the better initial layout from SA. The total cost remains unchanged from that of the initial layout regardless of the number of maximum allowable cuts Ncut_max. The root of the failure is tracked back to the poor performance of BD that is solutions from initial iterations oscillate widely from one region of the feasible set to another; thereby slowing the convergence rate of BD (Hiriart-Urruty and Lemaréchal[7]). In this case, the number of maximum allowable cut is only 500 negligibly small compared to the total number of cuts of the problem $((n!)^T = (20!)^5)$. Consequently, ABD poorly samples layouts from the feasible space for ADP. The TRC restricting the change of the solution from one to the next iteration should improve the sampling process of ABD and is investigated later.

Figure 1 shows the dynamics of ABD+ADP. ADP in every iterations except the last one significantly reduces the total cost from ABD. On the contrary, ABD in every iterations except the first one fails to reduce the cost from ADP in the previous iteration. Evidently, the performance of ABD+ADP is hinged on the solution from ADP not that from ABD. Nevertheless, the quality of ADP solution implicitly relies on sample layouts generated from ABD.

As Ncut_max increased, the total cost improves and the executable time increases (Table 2). The improvement of the total cost is most likely to be a result of the increase in the number of sample layouts for ADP from ABD.

TRC: TRC takes advantage of the sub-optimal layout from SA and improves the sub-optimal solution (see Table 1). The total cost reduces to 4,360,788. The TRC restricts the change of solution during each cut in ABD such that the new solution is in the neighbor of the previous solution. In this way, the TRC allows thoroughly search for the best solution in the neighbor of the previous sub-optimal solution; thereby enhancing the performance of TRC. Figure 2 reveals that the dynamics of TRC is similar to that of ABD+ADP.

Evidently from Table 1, TRC outperforms ABD+ADP in all aspects-the quality of the total cost and the executable time. In particular, the executable time of TRC is almost three order of magnitudes lower than ABD+ADP with Ncut_max = 500. TRC outperforms ABD+ADP for n = 40, T = 3 problem as shown in Table 1.

SAP: The total cost from the procedure starting from SA sub-optimal initial layout shown in Table 1 is 4,316,387 and the grand total cost reduction over the whole procedure is 1.5%. SAP produces a better sub-optimal solution than TRC and a comparable sub-optimal solution compared with other metaheuristic methods. The most two effective steps produce the cost reduction of 0.542 and 0.469% respectively as shown in Fig. 3 occurring at Ncut_max of 40 and TR_size_constant of 0.25 and 0.5 respectively. The most two effect steps indeed account for about 2/3 of the grand total cost reduction. Unlike the first few steps running at Ncut_max of 10 and 20, they seem not to improve the solution so much. With a good initial layout, SAP requires sufficient Ncut_max before it can produce significant improvement in the total cost when the trust-region size is relatively restricted. Also, there is not much improvement at Ncut_max of 80. It is speculated that with this even better solution, the procedure requires more Ncut_max than 80. The dynamics of SAP over each TRC is similar to a single TRC and ABD+ADP as shown in Fig. 4. SAP takes less time than ABD+ADP but longer than TRC. Nevertheless, its sub-optimal solution is better than ABD+ADP's and TRC's (Table 1).

With an arbitrary initial layout, the most cost reduction accounting almost all of the total reduction occurs during Ncut_max of 10 and TR_size_constant of 1 as shown in Fig. 5. The convergence behavior in this

case is much more organized than SAP starting from a sub-optimal initial layout. A total cost is significantly reduced during low Ncut_max for a large TR_size_constant and its reduction decrease as TR_size_constant decreases. The total cost reduction significantly reduces again when Ncut_max increases even though the total cost reduction is not as large as the previous steps. Evidently, SAP takes a great advantage of ADP to single out the best sub-optimal solution from random layouts generated by ABD with a large TR_size_constant during initial steps. With the reduction of TR_size_constant further, ABD tends to sample layout only in the neighbor of a sub-optimal layout from the previous step. As a result ADP slightly improves the solution from the previous step. If the true global minimum lays further way from this neighbor, SAP may not be able to single out the true global minimum.

CONCLUSION

New combinatorial solution methods for large-scale DQAPs are developed based on an equivalent linear problem. The principle algorithms for the proposed combinatorial method are based on BD successfully applied to large-scale MILP problem and ADP successfully applied to dynamic assignment problem. Due to slow convergence of BD and limited ability of a MILP algorithm, BD is relaxed by linear programming and Hungarian algorithm. The methods are further accelerated by the TRC and the SAP. In summary, the overall performances of the proposed methods are comparable with other metaheuristic methods.

REFERENCES

1. Lawler, E.L., 1963. The quadratic assignment problem. Manage. Sci., 9: 586-599. DOI: 10.1287/mnsc.9.4.586
2. Benders, J.F., 1962. Partitioning procedures for solving mixed variables programming problems. Num. Math., 4: 238-252. DOI: 10.1007/BF01386316
3. Santoso, T., S. Ahmed, M. Goetschalckx and A. Shapiro, 2005. A Stochastic programming approach to supply chain network design under uncertainty. Eur. J. Operat. Res., 167: 96-115. http://cat.inist.fr/?aModele=afficheN&cpsidt=16876124
4. Rosenblatt, M.J., 1986. The dynamics of plant layout. Manage. Sci., 32: 76-82. DOI: 10.1287/mnsc.32.1.76
5. Luangpaiboon, P., 1995. Dynamic process layout planning. Master Thesis, Kasetsart University. http://intanin.lib.ku.ac.th/search*thx/aPongchanun+Luangpaiboon/apongchanun+luangpaiboon/-3%2C-1%2C0%2CB/frameset&FF=apongchanun+luangpaiboon&1%2C1%2C
6. Muenvanichakul, S., 1998. An hybrid approach of genetic algorithm/ simulated annealing and tabu search method for dynamic process layout planning. Master Thesis, Kasetsart University. http://intanin.lib.ku.ac.th/search*thx/aSirirat+Muenvanichakul/asirirat+muenvanichakul/-3%2C-1%2C0%2CB/frameset&FF=asirirat+muenvanichakul&1%2C1%2C
7. Hiriart-Urruty, J.B. and C. Lemaréchal, 1996. Convex Analysis and Minimization Algorithms: Part 2: Advanced Theory and Bundle Methods. Corrected End., Springer, ISBN: 3540568506, pp: 347.