Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

Parallel Computing 29 (2003) 631–639

PARALLEL
COMPUTING

www.elsevier.com/locate/parco

Parallel tabu search for a pickup and delivery problem under track contention

Pierpaolo Caricato [a], Gianpaolo Ghiani [b,*], Antonio Grieco [b], Emanuela Guerriero [a]

[a] *Dipartimento di Elettronica, Informatica e Sistemistica, Università degli Studi della Calabria, 87030 Rende (CS), Italy*

[b] *Dipartimento di Ingegneria dell'Innovazione, Università degli Studi di Lecce, 73100 Lecce, Italy*

Received 17 June 2002; accepted 28 October 2002

Abstract

This article introduces the *Pickup and Delivery Problem under Track Contention*, a particular vehicle routing problem in which loads have to be transported between origin–destination pairs by means of vehicles travelling along a capacitated network. Two sequential heuristics and a parallel tabu search are proposed. Computational experiments show that the parallel tabu search is able to find much better solutions than the sequential procedures, although this comes at the expense of a higher computing time.

Keywords: Pickup and delivery problem; Parallel computing; Metaheuristics

1. Introduction

Vehicle routing problems (VRPs) are central to logistics management. They consist of determining optimal vehicle routes through a set of users, subject to side constraints. A special class of VRPs arises in automated warehouses, flexible manufacturing systems and port terminals where *automated guided vehicles* (AGVs) are often

* Corresponding author.
E-mail addresses: pierpaolo.caricato@unile.it (P. Caricato), gianpaolo.ghiani@unile.it (G. Ghiani), antonio.grieco@unile.it (A. Grieco), emanuela.guerriero@unile.it (E. Guerriero).

0167-8191/03/$ - see front matter
doi:10.1016/S0167-8191(03)00046-2

used to move loads between workstations [7,9,10,13]. In such a context, the travel network consists of a physical or virtual path layout made up of *tracks*. Each track has a unit capacity, i.e., it can be traversed by a single vehicle at a time. When two or more vehicles try to enter a track at the same time, a *contention* arises and at least one of them will be delayed or re-routed.

While VRPs related to distribution management have been examined extensively [12], the literature on VRPs under Track Contention is fairly less developed and quite disorganized [8]. In most papers (see, e.g., [2] and [3]), a hierarchical approach is used:

(i) loads are dispatched to vehicles without any concern for track contentions;
(ii) track contentions are then detected and simple rules are used to decide in which order vehicles have to access a contended track.

For instance, priority can be given to vehicles according to a round-robin policy, or according to the delay accumulated in previous track contentions.

In most real-world VRPs with Track Contention, requests are revealed during the planning horizon and vehicle routes and schedules have to be built in an on-going fashion. In this paper we deal with a static problem (i.e., a problem where all requests are known in advance) that captures most aspects of problems arising in practice. Clearly, the management of a dynamic setting asks for a certain number of adjustments to our current problem solving methodology which are left as a future research work. The paper is organized along the following lines. In Section 2, a Pickup and Delivery Problem under Track Contention is presented, while in Section 3 three heuristics are described. Then a parallel implementation is presented in Section 4. Finally, computational results are reported in Section 5, followed by conclusions in Section 6.

2. Problem definition

The Pickup and Delivery Problem under Track Contention (PDPTC) can be defined as follows. Let $G = (V, E)$ be an undirected graph, where V is a vertex set and E is a set of edges (v_i, v_j). In our modelling, edges and vertices represent tracks and track intersections, respectively, while workstations are associated to a subset of vertices $V_1 \subseteq V$ (Fig. 1). With each edge (v_i, v_j) is associated a traversal time t_{ij}. As in real-world problems all tracks have the same length [8], it is assumed that $t_{ij} = 1$. Let T_{ij} be the duration of a shortest path from $v_i \in V_1$ to $v_j \in V_1$. A set $R = \{r = (a_r, b_r) : a_r \in V_1, b_r \in V_1\}$ of unit demand loads have to be transported between pairs of workstations by a homogeneous fleet of *m unit capacity* vehicles. Vehicles may be based at the same depot v_0 (as assumed in the following), or, as customary in Material Handling Systems, each vehicle may be based at a different depot. Each vehicle occupies an edge at any time, while each edge can accommodate at most one vehicle at a time. Our problem is to design a set of m vehicle routes and schedules in such a way that:

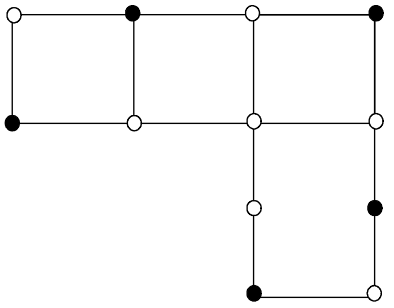

Fig. 1. Path layout (black vertices are in V_1).

(i) each request is serviced by exactly one vehicle;
(ii) each vehicle services one request at a time;
(iii) the route of a vehicle starts and ends at v_0;
(iv) the duration of the longest route (usually called *makespan*) is minimized.

It is worth noting that if the tracks are uncapacitated, the problem above reduces to an *m-Traveling Salesman Problem with Minmax Objective* (Minmax m-TSP) [4] as follows. Let $G' = (R \cup \{v_0\}, A')$ be a directed graph, where $A' = \{(r,s) : r \in R \cup \{v_0\}, s \in R \cup \{v_0\}, r \neq s\}$. To each arc $(r,s) \in A'$ is associated a traversal time.

- $T'_{rs} = T_{a_r b_r} + T_{b_r a_s}$ if $r, s \in R$;
- $T'_{rs} = T_{v_0 a_s}$ if $r = v_0$ and $s \in R$;
- $T'_{rs} = T_{a_r b_r} + T_{b_r v_0}$ if $r \in R$ and $s = v_0$.

There are m vehicles based at v_0. The m-Traveling Salesman Problem with Minmax Objective consists of determining m vehicle routes, starting and ending at v_0, and such that every vertex $r \in R$ is visited exactly once by one vehicle. The objective to be minimized is the duration of the longest route. França et al. [4] develop a tabu search heuristic and two exact search schemes. They are able to solve to optimality problems with up to 50 vertices. Given a Minmax m-TSP feasible solution x, a feasible PDPTC solution ξ can be obtained as follows:

(i) every arc (r,s) of the Minmax m-TSP feasible solution x is substituted for a path between a_r and b_r, followed by a path between b_r and a_s;
(ii) vehicles are then scheduled along the routes in such a way that track contention constraints are satisfied.

3. Heuristics

Our problem is strongly $\mathcal{NP}$-hard as it is a generalization of the Minmax m-TSP, as shown in Section 2. At present there is little hope to solve this problem to optimality

even for moderately sized instances. In what follows, three heuristics are illustrated.

3.1. Constructive heuristic

The first heuristic determines a feasible solution x to the associated Minmax m-TSP by means of the tabu search procedure described in [4], then converts it into a feasible solution ξ. The planning horizon $[0,+\infty)$ is divided into intervals $[0,D-1]$, $[D,2D-1]$, ... each having duration D. At the beginning of the hth interval, vehicles have been routed and scheduled until time $hD-1$ in such a way that track contention constraints are satisfied. Let $\tau_k(hD)$ be the duration of the kth vehicle schedule of ξ in case no track contentions arise in interval $[hD,\infty]$. In particular, $\tau_k(0)$ is the duration of the kth route of x. Phases (i) and (ii) are performed together as follows.

Make_feasible (x) procedure

Step 0 Set $h=0$.
Step 1 Sort the routes of x which are not finished at time hD by non-increasing values of $\tau_k(hD)$. If all routes have been finished, STOP.
Step 2 Schedule one route at a time in interval $[hD,(h+1)D-1]$.
Step 3 Set $h:=h+1$ and go to Step 1.

Step 2 is performed by using a modification of the A^* algorithm [11] which is a fast heuristic originally devised for finding shortest paths in a plane with moving obstacles. In our implementation, moving obstacles are represented by vehicles already routed and scheduled in interval $[0,hD-1]$.

3.2. Local search heuristic

This procedure determines a feasible solution x to the Minmax m-TSP by means of the tabu search heuristic described in [4]. Then, leaving unchanged the order in which the requests are serviced along the routes, the heuristic tries to reduce the makespan by re-routing or re-scheduling the vehicles. To this purpose a local search is performed by using the constructive heuristic as a subroutine. Let $\rho_1,\ldots,\rho_m$ be a permutation of the vehicle routes of x. Define the neighbourhood of $\rho_1,\ldots,\rho_m$ as the set of permutations that can be obtained by swapping a pair of routes.

Step 0 Choose an initial permutation $\rho_1,\ldots,\rho_m$ in such a way that $\tau_{\rho_1}(0)\geqslant\tau_{\rho_2}(0)\geqslant\cdots\geqslant\tau_{\rho_m}(0)$.
Step 1 Evaluate the neighbours of $\rho_1,\ldots,\rho_m$ by means of the *Make_feasible procedure*.
Step 2 If an improved permutation has been found, up-date the current permutation $\rho_1,\ldots,\rho_m$ and go to Step 1. Otherwise, STOP.

3.3. Tailored tabu search heuristic

Tabu search (TS) is a local search procedure that iteratively moves from a current solution to its best neighbour even if this causes a deterioration in the objective function value [6]. To avoid cycling a short term memory (like a tabu list or tabu attributes) is used. In addition, a diversification and an intensification phase can be used to visit new regions of the solution space, and to explore the more promising regions, respectively.

Our algorithm is a modification of the tabu search heuristic described in [4]. França et al. TS procedure [4] makes use of the GENI insertion heuristic and the US post-optimization procedure developed by Gendreau et al. [5]. An initial solution is first constructed by means of GENI. Neighbour solutions are then generated by repeatedly removing a vertex from its current route and inserting it using GENI into another route containing some of its closest neighbours. Then, the US post-optimization procedure is applied to each route separately. A parameter α determines how large the neighbourhood of the current solution is. Our tabu search heuristic has the same structure, except that for each Minmax m-TSP neighbour x a least makespan PDPTC solution is looked for by means of the local search procedure illustrated in Section 3.2.

4. Parallel implementation

In view of the adaptation of our findings to a real-time setting, we have developed a parallelization strategy for the tailored TS heuristic. This task can be accomplished in a number of ways depending on problem structure and hardware at hand. Crainic et al. [1] classify parallel TS procedures according to three criteria: *Search Control Cardinality*, *Search Control Type* and *Search Differentiation*. The first criterion indicates whether the control of the parallel search is performed by a single processor (*1-control*, 1-C), or distributed among several processors (*p-control*, p-C). The second measure denotes the way (*synchronous* or *asynchronous*) communication is performed among processors. In asynchronous communication, a processor that finds a new best solution broadcasts a message to the other processors. In the simplest case, the single solution is sent (*collegial* communication, C) while in *knowledge-based collegial* communication (KC) additional information are transmitted to the receivers. Finally, the third criterion accounts for the way different searches are carried on. Four alternatives are available: *single initial point-single strategy* (SPSS) if a single search is performed; *single initial point-multiple strategies* (SPMS) when each processor carries on a different search starting from the same initial solution; *multiple initial points-single strategy* (MPSS) if each processor performs the same search starting from different initial solutions; *multiple initial points-multiple strategies* (MPMS) if each processor carries on a different search starting from a different initial solution. In SPMS and MPMS, the searches may be performed by different algorithms or, more commonly, by the same algorithm with different parameters.

We have used a multi-thread search strategy in which p several distinct search paths are followed in parallel. Once a processor finds a new best solution, it sends

it to the other processors that re-initialise their searches. This approach should guarantee a high level of search diversification [1]. Each thread is characterized by a different value of parameter α. Moreover, in order to speed up the neighbourhood evaluation, which is by far the most time-consuming part of the algorithm, we have used a master–slave scheme for a each thread. The master process manages the França et al. TS main operations while p slaves perform the local searches.

5. Computational results

Both heuristics were coded in C++. The first two procedures were run on a standalone PC with a Pentium III processor clocked at 500 Mhz. The third algorithm was implemented on a cluster of four PCs, each of which has two Pentium III processors clocked at 700 Mhz. Each process was coded in C++ and process communications were handled by the *message passing interface* (MPI) software [14]. After preliminary tests, we have chosen $p = 4$ and $\alpha = 0.5, 1.0, 1.5, 2.0$. The heuristics were tested on a set of instances generated as follows. First, a 5×4 grid network $G_B = (V_B, E_B)$ (a *bay*) was generated. Then edges $(v_i, v_j) \in V_B$ were deleted with probability 0.5 in order to make G_B sparse. Finally, three graphs $G = (V, E)$ were built by linking together 2, 3, 4 copies of $G_B = (V_B, E_B)$ through single arcs (Fig. 2). After preliminary tests, we set $D = 10$. For each graph G we generated 20, 30, 40, 50 requests at random and for each set of requests we allowed $m = 3, 4, 5$ vehicles.

Computational results are reported in Tables 1–3. The meanings of the column headings not yet introduced are as follows:

- CH: constructive heuristic solution value;
- CH SEC: number of seconds required to run the constructive heuristic;
- LS: local search heuristic solution value;
- LS Saving: percentage saving of the local search heuristic solution value compared to the constructive heuristic solution value;
- LS SEC: number of seconds required to run the local search heuristic;
- TS: tabu search heuristic solution value;
- TS Saving: percentage saving of the tabu search heuristic solution value compared to the constructive heuristic solution value;
- TS SEC: number of seconds required to run the parallel tabu search heuristic.

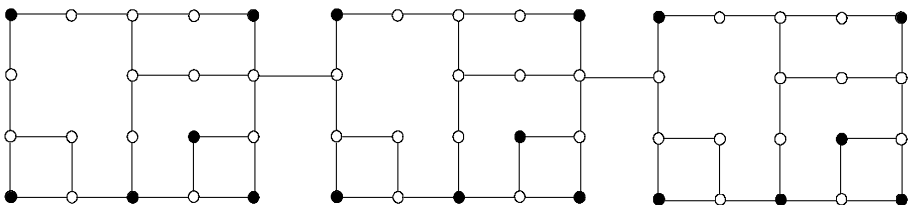

Fig. 2. Three bay graph.

Table 1
Computational results—two bay instances

m	$\|R\|$	CH	CH SEC	LS	LS Saving	LS SEC	TS	TS Saving	TS SEC
3	20	64	0.7	64	0	8.1	61	5	43
	30	103	1.6	103	0	17.2	98	5	126
	40	132	3.2	124	6	22.1	115	13	167
	50	220	5.4	219	0	42.1	178	19	323
4	20	63	1.3	55	13	6.9	52	17	98
	30	94	2.3	86	9	7.4	82	13	175
	40	180	6.5	146	19	54.3	102	43	318
	50	210	14.8	171	19	165.6	137	35	918
5	20	71	2.5	54	24	32.1	50	30	175
	30	106	12.5	94	11	112.1	80	25	579
	40	172	18.1	166	3	133.2	109	37	841
	50	237	29.8	213	10	152.3	163	31	1493

Table 2
Computational results—three bay instances

m	$\|R\|$	CH	CH SEC	LS	LS Saving	LS SEC	TS	TS Saving	TS SEC
3	20	75	0.7	62	17	11.7	62	17	82
	30	133	3.5	114	14	23.1	111	17	222
	40	245	13.6	245	0	99.4	178	27	605
	50	305	31.8	304	0	65.9	259	15	775
4	20	82	4.9	82	0	43.1	60	27	244
	30	155	16.2	142	8	127.5	93	40	704
	40	225	27.5	223	1	174.3	148	34	978
	50	176	26.3	171	3	143.2	166	6	799
5	20	68	12.3	63	7	62.3	57	16	432
	30	144	35.2	126	13	183.7	85	41	1330
	40	127	48.9	122	4	286.3	122	4	2045
	50	222	54.2	193	13	303.7	159	28	2232

Computational results show that the tabu search procedure is able to improve remarkably (22.7%, 22.5% and 26.9% on the average in the 2, 3 and 4 bay case, respectively) the solutions generated by the constructive heuristic although this usually comes at the expense of a larger computing time.

6. Conclusion

In this paper we have examined a particular VRP under Track Contention whose applications arise in automated Material Handling Systems and Flexible Manufacturing Systems. Three heuristics were developed: two simple procedures and a tailored tabu search. In view of its adaptation to a real-time setting, the tabu search

Table 3
Computational results—four bay instances

m	$\|R\|$	CH	CH SEC	LS	LS Saving	LS SEC	TS	TS Saving	TS SEC
3	20	81	2.1	81	0	23.1	76	6	139
	30	348	7.6	346	1	48.2	122	65	454
	40	202	9.2	202	0	53.4	146	28	560
	50	263	14.3	252	4	87.3	199	24	871
4	20	57	3.2	57	0	34.2	56	2	232
	30	201	9.8	177	12	78.3	87	57	672
	40	134	15.6	134	0	158.2	107	20	890
	50	253	42.3	246	3	430.1	177	30	2088
5	20	57	7.6	57	0	50.1	49	14	433
	30	87	17.2	85	2	124.8	77	11	940
	40	194	41.1	193	1	239.1	113	42	1577
	50	195	42.3	175	10	226.0	148	24	1493

algorithm was parallelized. Computational results show that this procedure is able to provide better solutions although this usually comes at the expense of a larger computing time. We are unaware of an other research that has raised the issue of track contention in vehicle routing.

References

[1] T.G. Crainic, M. Toulouse, M. Gendreau, Towards a taxonomy of parallel tabu search algorithm, INFORMS Journal on Computing 9 (1997) 61–72.
[2] P.J. Egbelu, J.M.A. Tanchoco, Characterization of automated guided vehicle dispatching rules, International Journal of Production Research 22 (1984) 359–374.
[3] J.J. Evers, S.A. Koppers, Automated guided vehicle traffic control at a container terminal, Transportation Research 30 (1996) 21–34.
[4] P.M. França, M. Gendreau, G. Laporte, F.M. Müller, The *m*-traveling salesman problem with minmax objective, Transportation Research 29 (1995) 267–275.
[5] M. Gendreau, A. Hertz, G. Laporte, New insertion and post-optimization procedures for the traveling salesman problem, Operations Research 40 (1992) 1086–1094.
[6] F. Glover, Future paths for integer programming and links to artificial intelligence, Computers and Operations Research 5 (1986) 533–549.
[7] K.H. Kim, J.W. Bae, Dispatching automated guided vehicles for multiple container-cranes. in: Proceedings of the Conference on Port Design and Operations Technology, Singapore, 4–5 November, 1999.
[8] Q. Ling, H. Wen-Jing, Scheduling and routing algorithms for AGVs: a survey, technical report CAIS-TR-99-26, Center for Advanced Information Systems, School of Applied Science, Nanyang Technological University, Nanyang, 1999.
[9] A. Langevin, D. Lauzon, D. Riopel, Dispatching, routing and scheduling of two automated guided vehicles in a flexible manufacturing system, International Journal of Flexible Manufacturing Systems 8 (1996) 246–262.
[10] Q. Ling, H. Wen-Jing, Scheduling of Automated Guided Vehicles in a Mesh-like Topology: a Case Study in a Container Terminal, Technical report CAIS-TR-01-35, Center for Advanced Information Systems, School of Applied Science, Nanyang Technological University, Nanyang, Singapore, 2001 (http://www.cais.edu.sg:8000/).

[11] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, Englewood Cliffs, NJ, 1995.
[12] P. Toth, D. Vigo (Eds.), The vehicle routing problem, Monographs on Discrete Mathematics and Applications, SIAM, Philadelphia, 2002.
[13] V.Y. Vee, R. Ye, S.N. Shah, W.J. Hsu, Meeting Challenges of Container Port Operations for Next Millennium. Technical Report CAIS-TR-99-25, Centre for Advanced Information Systems, Nanyang Technological University, Singapore, 1999 (http://www.cais.edu.sg:8000/).
[14] G. William, M. Snir, W. Gropp, B. Nitzberg, E. Lusk, MPI: The Complete Reference, MIT Press, Boston, Massachussetts, 1998.