Time Optimized Multi-Agent Path Planning using Guided Iterative Prioritized Planning

(Extended Abstract)

Wenjie Wang
Nanyang Technological University
School of Computer Engineering, Singapore
wang0570@e.ntu.edu.sg

Wooi Boon Goh
Nanyang Technological University
School of Computer Engineering, Singapore
aswbgoh@ntu.edu.sg

ABSTRACT
This paper proposes the guided iterative prioritized planning (GIPP) algorithm to address the problem of moving multiple mobile agents to their respective destinations in a shortest time-related cost. Compared to other MAPP algorithms, the GIPP algorithm strikes a good balance between various performance criteria such as finding feasible solutions, completing the task promptly and low computational cost.

Categories and Subject Descriptors
I.2.9 [**Artificial Intelligence**]: Robotics – *Workcell organization and planning.*

General Terms
Algorithms, Experimentation.

Keywords
Path planning, Multi-agents, Guided local search

1. INTRODUCTION
Many MAPP algorithms do not scale well when increasing numbers of agents are required to move concurrently. As such, finding an optimal solution under these conditions can be computationally expensive. Many MAPP approaches make a trade-off between the solution optimality and the computational complexity of the algorithm. Coupled approaches combine the workspace of all agents into a composite workspace, and then plan their paths simultaneously. They can find an optimal solution, but are not computationally scalable. The alternative decoupled approaches [1], [2], as used in this work, decompose the MAPP problem into several sub-problems and then solve these sub-problems separately. These approaches are notorious for producing inferior solutions or poor performance when the environment is only mildly crowded. This paper defines the time-optimized MAPP problem and then proposes the GIPP algorithm that is able to find a good solution for the defined problem in crowded environments.

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*, Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.

2. PROBLEM DEFINITION
The time-optimized MAPP problem is described as follow: each agent A_i, $i \in \{1,..,n\}$, is required to move concurrently along a collision-free path P_i from a unique starting point S_i to its destination G_i in a shortest time. Assume the path node $P_i(t)=(P_i^x(t),P_i^y(t))$ denotes the position of each agent A_i at the time step t, and each agent stops at its destination at the time step T_i. The function $f_i(P)$ calculates the time taken by agent A_i to reach its destination safely with respect to the paths of the other $(n-1)$ agents. The function $\varphi_i(t)$ denotes whether the agent A_i collides with the static obstacles S at the path node $P_i(t)$. The function $\Omega_{i,j}(t)$ denotes whether the agent A_i collides with the agent A_j at the path node $P_i(t)$. The function $\phi_{i,j}(t-1,t)$ indicates whether there is any head-on collision between agent A_i and A_j during the time step $t-1$ to t. The function $\varphi_i(t)$, $\Omega_{i,j}(t)$ or $\phi_{i,j}(t-1,t)$ will be set to 1 if its corresponding statement is true, or else set to 0. As a result, our time-optimized objective function is described as follows:

$$min\ F(P) = max(f_1(P), \dots, f_n(P)) \text{ subjected to} \quad (1)$$

$$\varphi_i(t)=0 \text{ for each } \{i,t\}, \quad (2a)$$

$$\Omega_{i,j}(t)=0 \text{ and } \phi_{i,j}(t-1,t)=0 \text{ for each } \{i,j,t\} \quad (2b)$$

Where $i \in \{1,..,n\}$, $j \in \{1,..,n\}$, $j \neq i$, $t \in \{1,..,T\}$, $T = max_{1 \le i \le n}(T_i)$. The constraints in (2a) and (2b) are called static constraint and dynamic constraints respectively.

3. GUIDED ITERATIVE PRIORITIZED PLANNING ALGORITHM
The guided iterative prioritized planning (GIPP) algorithm (pseudo-code is given in Figure 1) is a local search method that iteratively finds an improving solution from a defined neighborhood $N(x)$ of the current solution x with respect to a cost function. Using an efficient optimal search algorithm like the A* algorithm for single agent path planning, we can define n different neighborhood $N_i(P)$ obtained by applying changes in one component of the current solution $P=(P_1,..,P_n)$. For each local neighborhood $N_i(P)$, we adopt a corresponding local cost function $F_i(P;\mu)$ given by:

$$min\ F_i(P;\mu) = f_i(P) + \mu * \sum_{j=1,j\neq i}^{n} \sum_{t=1}^{T_i} \{h_j(P) * [\Omega_{i,j}(t) + \phi_{i,j}(t-1,t)]\} \quad (i \in \{1,..,n\}) \quad (3)$$

Subjected to (2a). Where μ is a positive penalty coefficient, and h_j is a penalty associated with a defined solution feature c_j and initially set to 1. We define n solution features that correspond to the path of each agent. Each solution feature c_i for $i \in \{1,.,n\}$ is defined as the violation of the dynamic constraints imposed by path P_i. The utility function of each solution feature c_i is given by

$$U(P^*, i) = \sum_{j=1, j\neq i}^{n} \sum_{t=1}^{T_i} \{h_j(P) * [\Omega_{i,j}(t) + \phi_{i,j}(t-1, t)]\} \quad (4)$$

where $D_i(P^*)$ indicates whether the feature c_i is present in the solution P^*. $D_i(P^*) = 1$ means it does, or else $D_i(P^*) = 0$.

When a local optimal solution P^* is encountered, a utility for each solution feature c_i is calculated according to function (4). Then $h_i(P^*)$ of the solution feature c_i with the highest utility present in the solution P^* is increased. The local optimal solution P^* is deemed to be reached when the current solution is continuously kept for at least *n* iterations, where *n* is the number of agents.

1) Generate an initial solution $P^{(0)}=(P_1^{(0)},..., P_n^{(0)})$ and set $i \leftarrow 1$, $\theta \leftarrow 0$, $\tau \leftarrow 1$, and set a high value for μ;
2) Repeat the following procedure until some stopping criteria are satisfied:
   a) At the iteration τ, try to find an improving solution P' with respect to the cost function $F_i(P; \mu)$ in $N_i(P^{(\tau-1)})$;
   b) $P^{(\tau)} \leftarrow P'$;
   c) If $P^{(\tau)} = P^{(\tau-1)}$, $\theta \leftarrow \theta+1$, or else $\theta \leftarrow 0$;
   d) If $\theta \geq n$, $\theta \leftarrow 0$, calculate the utility $U(P^{(\tau)}, j)$ of each feature c_j for $j=1,2,..,n$ and then increase the value of the feature penalty h_j of the feature with the maximal utility by a value of ε.
   e) $\tau = \tau + 1$, then $i = 1$ if $i \geq n$, or else $i = i + 1$;

Figure. 1 Guided iterative prioritized planning for MAPP

The iteration stops when there is no further improvement with the current feasible solution or an upper iteration bound δ is reached. Since the neighborhood $N_i(P^{(\tau-1)})$ is obtained by applying changes in one component P_i at each iteration, we essentially have a simple single agent path planning problem with the incremental positions of other agents treated as obstacles. Cooperative A* [2] is used to find a path for each agent at each iteration.

4. EXPERIMENTAL RESULTS

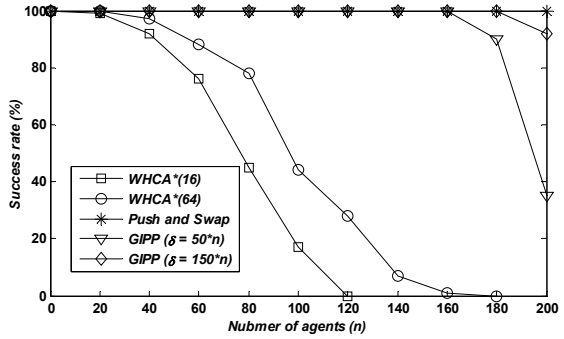

Figure 2. Success rates of windowed hierarchical cooperative A* with 16 window step (WHCA*(16)) and 64 window step (WHCA*(64)), the Push and Swap, and GIPP algorithms.

The GIPP algorithm was evaluated in a simulation consisting of a 30×20 grid map with 20% static obstacle nodes. Each run uses the same 100 random scenarios. The Visual C++ simulation code was executed on an Intel(R) core™ 2 CPU (2.67 GHz) with 2 GB memory. We compared our proposed GIPP algorithm with the Windowed Hierarchical Cooperative A*(WHCA*) [1] and Push and Swap [2] algorithms. In all the experiments presented, the settings for GIPP are μ=100 and ε = 1.

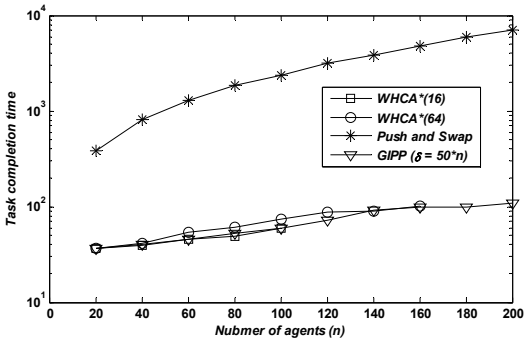

Figure 3. Comparative task completion time measures the average time steps to move *n* agents to their destinations.

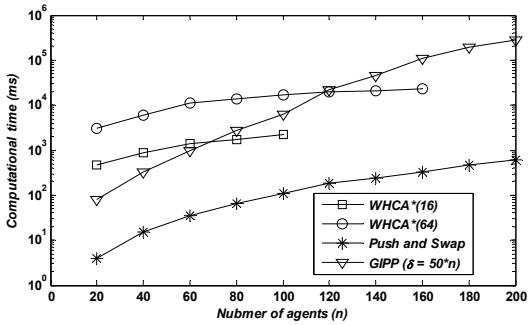

Figure 4. Comparative computational time measures the average time taken to compute viable paths for all *n* agents.

Though a larger window size allowed the WHCA*(64) algorithm to achieve better success rate (i.e. percentage of time in 100 scenarios all *n* agents found viable paths), it was still much lower than that of the GIPP algorithm. The computational cost of WHCA*(64) also increased with the increased window size (see Fig. 4). In contrast, the Push and Swap algorithm moves one agent at a time and does not suffer from head-on collisions. This increases its ability to find feasible solutions in crowded environments (see Fig. 2). However, moving one agent at a time cannot provide time-optimized MAPP solutions (see Fig. 3) resulting in significantly longer task completion time than the GIPP algorithm. GIPP can achieve comparable success rate to that of Push and Swap if more iterations are allowed. Unlike WHCA* and Push and Swap, the GIPP algorithm is designed with the ability to explore the whole solution space and therefore suffers from high computational cost (see Fig. 4). Fortunately, this drawback can be readily resolved by using faster computation hardware.

5. ACKNOWLEDGEMENTS

This research is supported by the Singapore National Research Foundation (NRF2008-IDM001-017) and administered by the Ministry of Education.

6. REFERENCES

[1] Silver, D. 2005. Cooperative Pathfinding. In Proceedings of AIIDE, 117-122.

[2] Luna, R. and Bekris, K. E. 2011. Push and Swap: Fast Cooperative Path-Finding with Completeness Guarantees, In Proceedings of IJCAI, 294-300.