Lagrangian Relaxation for Large-Scale Multi-Agent Planning

Geoffrey J. Gordon[†], Pradeep Varakantham[‡], William Yeoh[*], Hoong Chuin Lau[‡], Ajay S. Aravamudhan[‡] and Shih-Fen Cheng[‡]
[†]*Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA*
[‡]*School of Information Systems, Singapore Management University, Singapore 178902*
[*]*Computer Science Department, New Mexico State University, Las Cruces, NM 88003, USA*
[†]*ggordon@cs.cmu.edu* [‡]*{pradeepv,hclau,ajaysa,sfcheng}@smu.edu.sg* [*]*wyeoh@nmsu.edu*

***Abstract*—Multi-agent planning is a well-studied problem with various applications including disaster rescue, urban transportation and logistics, both for autonomous agents and for decision support to humans. Due to computational constraints, existing research typically focuses on one of two scenarios: unstructured domains with many agents where we are content with heuristic solutions, or domains with small numbers of agents or special structure where we can provide provably near-optimal solutions. By contrast, in this paper, we focus on providing provably near-optimal solutions for domains with large numbers of agents, by exploiting a common domain-general property: if individual agents each have limited influence on the overall solution quality, then we can take advantage of randomization and the resulting statistical concentration to show that each agent can safely plan based only on the *average* behavior of the other agents. To that end, we make two key contributions: (a) an algorithm, based on Lagrangian relaxation and randomized rounding, for solving multi-agent planning problems represented as large mixed-integer programs; (b) a proof of convergence of our algorithm to a near-optimal solution. We demonstrate the scalability of our approach with a large-scale illustrative theme park crowd management problem.**

I. Introduction

Rapid progress in ubiquitous computing has enabled real-time delivery of contextualized information via devices (such as mobile phones and car navigation devices) over wide areas. As a result, a new kind of information service for mass user support is beginning to emerge. Examples include services that coordinate movements of first responders during a disaster rescue [1], movements of taxis in a fleet [2] and movements of visitors in leisure destinations (such as theme parks or World expositions). In these services, users are typically represented by computational agents that perform real-time planning and adaptation. Designing coordination mechanisms that can govern these services in ways that meet global criteria such as fairness, revenue maximization, stability/convergence, and efficient resource utilization is a research challenge.

Motivated by this challenge, we consider in this paper a large-scale multi-agent planning problem both for autonomous agents and for decision support to humans. We seek provably near-optimal solutions for domains with large numbers of agents by exploiting *influence limits* for individual agents. Our method is based on *Lagrangian relaxation*, a technique that has often been successfully applied to solve centralized optimization problems, together with a straightforward *randomized rounding* method.

Lagrangian relaxation has the advantage in that it inherently produces subproblems that can be solved in parallel. This property fits very naturally in a multi-agent environment, where each agent reasons about its own subproblem. Our contributions in this paper are thus: (a) a new parallelizable algorithm for solving multi-agent planning problems, and (b) a proof of convergence to optimality as the number of agents increases. In fact, the quality of the solution actually *improves* as the problem size increases.

II. Background: Lagrangian Relaxation

Lagrangian relaxation [3] is a well known algorithm-design strategy for solving constrained optimization problems, where there are two types of constraints: (a) *local* constraints, which factor across agents, and (b) *global* constraints, which couple agents and make the problem difficult to solve. The first step is to dualize the global constraints by introducing Lagrange multipliers (referred to as *prices*). The global planning problem then splits into a number of *slave problems*, which have only local constraints, plus one *master problem*. The goal of the master problem is to determine the correct value for the vector of prices. Given these prices, the slave problems are completely *independent*. So, starting from an initial price vector, we can solve the slave problems in parallel, and then use the resulting solutions to improve the prices, e.g., by gradient descent. This procedure of updating the prices is continued until all the global constraints are satisfied.

Lagrangian relaxation has two key drawbacks that preclude its application in the domains of interest in this paper: (a) it does not provide any bounds on the quality of the solution, and (b) it can oscillate and fail to converge if the slave problems are too homogeneous. In this paper, we overcome both drawbacks, allowing Lagrangian relaxation to be used in large-scale multi-agent planning problems.

III. Illustrative Domain

We motivate our work with a theme park crowd management problem, where each patron needs to plan its route, that is, its sequence of attractions to visit, such that his total waiting time is minimized. We represent this problem with a tuple $\langle A, P, \{A(p_i)\}_1^n, \{d_{a_i}\}_1^k, \{U_i\}_1^n, H\rangle$, where $A = \{a_i\}_1^k$ is the set of attractions in the theme park; $P = \{p_i\}_1^n$ is the set of patrons in the theme park; $A(p_i) \subseteq A$ is the subset of attractions that patron p_i prefers to visit; d_{a_i} is the service rate of attraction a_i, that is, the number of patrons that attraction a_i can service in each time step; U_i is the utility function of patron p_i; and H is the horizon of the problem. The goal is to find the route π_i for each patron p_i such that the sum of utilities $U_i(\pi_i)$ over all patrons is maximized.

Fig. 1 shows a centralized mixed-integer program (MIP) formulation of the problem, where we discretize the queue at each attraction into buckets.

$_p c^t_{b,a}$ is the utility of patron p being in bucket b of attraction a at time step t. We set a utility of 10 when a patron is serviced at a preferred attraction, a utility of -1 when a patron is queuing up.

$_p x^t_{b,a}$ is the binary decision variable indicating whether patron p is at bucket b of attraction a at time step t. Constraint 2 ensures that the number of patrons at each bucket does not exceed its service rate. Constraint 3 ensures that each patron can be in at most one bucket at each time step, that is, he is in at most one queue position at each time step. Constraint 4 ensures that each patron can be serviced at most once by each attraction, that is, he does not repeatedly visit an attraction. Constraint 5 ensures that if a patron p exits the park at time step t, that is, when $\sum_{a,b} {_p x^t_{b,a}} = 0$, then he does not reenter the park at a future time step. Constraint 6 ensures that patrons do not jump queue, that is, they move to the next bucket in each time step.

Using the terminology from the "Background: Lagrangian Relaxation" section, Constraints 2 are the global constraints, while the others are local constraints.

$$\max \sum_{p,b,a,t} {_p x^t_{b,a}} \cdot {_p c^t_{b,a}} \qquad \text{s.t.} \tag{1}$$

$$\sum_{p} {_p x^t_{b,a}} \le d_a \qquad \forall b, a, t \tag{2}$$

$$\sum_{b,a} {_p x^t_{b,a}} \le 1 \qquad \forall p, t \tag{3}$$

$$\sum_{t} {_p x^t_{1,a}} \le 1 \qquad \forall p, a \tag{4}$$

$$\sum_{b,a} {_p x^t_{b,a}} \ge \sum_{b,a} {_p x^{t+1}_{b,a}} \qquad \forall p, t < H \tag{5}$$

$${_p x^t_{b,a}} \le {_p x^{t+1}_{b-1,a}} \qquad \forall p, b > 1, a, t < H \tag{6}$$

Figure 1. Centralized MIP Formulation

IV. Multi-Agent Planning Problem

In this paper we are interested in multi-agent planning problems where the interactions between agents are due to common resource consumption. For example, in our illustrative theme park crowd management problem, the resources are the buckets for which the agents contend (acting on behalf of humans). We represent such large-scale multi-agent planning problems as MIPs with special structure.

Our chief assumptions are: (i) *Factored structure*: apart from interactions due to common resource consumption, we assume that the planning problems of individual agents are independent of each other [4], [5]. However, our approach easily extends to cases with common global states (although we do not include this generalization due to space constraints). (ii) *Influence limit*: the influence of each agent on the overall solution quality and resource consumption is limited. (iii) *Existence of local planning subroutines*: our approach does not depend on the specific local planning problems (MIPs, MDPs, etc.), but we do assume that there exist tractable procedures to solve them.

A. Factored Structure

More formally, we suppose that agent i's plan is represented by a set of decision variables $x_i \in \Re^{n_i}$,[1] subject to *local* constraints $A_i x_i = b_i$, $x_i \in X_i$ and *local* costs $c_i^\top x_i$. These local terms represent restrictions (e.g., on movement) or rewards/costs (e.g., from accomplishing local goals) that are due to the actions of an individual agent. The agents interact through *coupling* constraints, based on production or consumption of *generalized resources*: resource j is described by a closed proper convex function $f_j : \Re \to \Re \cup \{\infty\}$ (which represents the global cost of consuming a certain quantity of the resource) an vectors $\ell_{ij} \in \Re^{n_i}$ (which govern the relationship between agent i's decision variables and the global consumption $y_j = \sum_{i=1}^n \ell_{ij}^\top x_i$).[2] The *global planning* problem of interest is therefore:

$$\max_x \; V_p(x) \;\; \text{s.t.} \;\; A_i x_i = b_i, \; x_i \in X_i \quad \forall i \tag{7}$$

$$V_p(x) = \sum_{i=1}^n c_i^\top x_i - \sum_{j=1}^m f_j\left(\sum_{i=1}^n \ell_{ij}^\top x_i\right)$$

Typical examples of resources include fuel, water, subassemblies of a final product, or the limitation that only one agent at a time can occupy a given location. By choosing f_j appropriately, we can obtain either soft or hard constraints: e.g., for a soft version of the constraint $y_j \le d_j$, we could set $f_j(y_j)$ to be the *hinge loss* function $\max(0, k(y_j - d_j))$,

[1] If each agent's planning problem is an MDP, then these would be associated with every state-action pair.

[2] Negative consumption (i.e., production) is allowed.

where $k > 0$ determines the strength of the soft constraint. For a hard constraint, we could set

$$f_j(y_j) = \begin{cases} 0 & y_j \leq d_j \\ \infty & y_j > d_j \end{cases}$$

(equivalent to setting $k = \infty$ in the hinge loss).

1) Piecewise Linear Resource Cost:: For computational simplicity, we assume that each resource cost function f_j is piecewise linear: that is,

$$f_j(y_j) = \max\{\alpha y_j + \beta \mid (\alpha, \beta) \in F_j\}$$

where F_j is a finite set of slope-intercept pairs (α, β). The assumption of piecewise linearity loses us very little: for a soft constraint, the hinge loss $\max(0, k(y_j - d_j))$ is clearly piecewise linear, with $F_j = \{(0,0), (k, -kd_j)\}$. For a hard constraint, we can use the well-known trick of substituting a hinge loss with a sufficiently large finite slope k. Finally, to simulate a smooth f_j, we can use a large number of pieces to achieve any desired approximation accuracy.

B. Influence Limit

As described so far, the global planning problem (Eq. 7) is *NP*-hard and inapproximable to within any constant factor (unless $P = NP$). We thus cannot in general hope to make progress without further restrictions. We propose that a natural restriction is an *influence limit* for each agent. Intuitively, such a limit captures the idea that each agent has a bounded effect on the quality of the overall solution. The influence limit has two parts. First, no agent controls a disproportionate share of the utility or resources: writing V_p^* for the optimal value in Eq. 7, and y_j^* for the usage of resource j in some optimal solution, we assume that there is a constant $U > 0$ such that

$$-\frac{U}{n}|V_p^*| \leq c_i^\top x_i \leq \frac{U}{n}|V_p^*| \tag{8}$$

$$-\frac{U}{n}|y_j^*| \leq \ell_{ij}^\top x_i \leq \frac{U}{n}|y_j^*| \tag{9}$$

for all i, j, and $x_i \in X_i$.

Second, we suppose that the optimization problem as a whole is well conditioned, so that a small change in resource availability leads to only a small change in solution quality. In particular, we consider redefining the consumption cost for resource j in Eq. 7 to be

$$f_j\left(\epsilon_j + \textstyle\sum_{i=1}^n \ell_{ij}^\top x_i\right) \tag{10}$$

for some small $\epsilon_j \geq 0$. Let V_ϵ^* be the optimal value of Eq. 7 after substituting in the definition of Eq. 10 for all j.

Given this definition, we assume that there exists a bound $\epsilon_{\max} > 0$, a condition number $\kappa > 0$, and a discretization level $\Delta > 0$ such that

$$V_\epsilon^* \geq V_p^* - \kappa \textstyle\sum_j \epsilon_j - \Delta/n \tag{11}$$

whenever $0 \leq \epsilon_j \leq \epsilon_{\max}, \forall j$. This restriction eliminates problems that are balanced at the knife edge of feasibility. In more detail, this assumption says that (within the range $[0, \epsilon_{\max}]$) a decrease in resource availability has two effects: first, we lose utility in proportion to the loss of resources, at a rate no greater than κ. Second, there can be a discrete loss in utility, independent of ϵ_j, due to integrality constraints: e.g., changing $x \leq 4.01$ to $x \leq 3.99$ means that the largest feasible integer x changes from 4 to 3. This discrete loss must get smaller as we increase n: proportionally, the effect of going from $x \leq 400.01$ to $x \leq 399.99$ must be smaller than for changing from 4.01 to 3.99.

V. Subgradient Lagrangian Relaxation

We now describe our Lagrangian-relaxation-based algorithm for our multi-agent planning problem, that is, the global planning problem of Eq. 7 along with its modification in Eq. 10. Not only is the algorithm efficient, it is also able to converge in the presence of homogeneous agents to a solution with quality guarantees.

Lagrangian relaxation of the global constraints yields:

$$\begin{aligned} V_d^* &= \inf_\lambda \Big\{ \max_x \Big\{ \sum_{i=1}^n c_i^\top x_i - \sum_{j=1}^m \Big[\lambda_j y_j - f_j^*(\lambda_j)\Big]\Big\}\Big\} \text{ s.t. } \mathcal{C} \\ &= \inf_\lambda \Big\{ \sum_{j=1}^m f_j^*(\lambda_j) + \sum_{i=1}^n \max_{x_i} \Big[c_i^\top x_i - \sum_{j=1}^m \lambda_j y_j\Big]\Big\} \text{ s.t. } \mathcal{C} \end{aligned} \tag{12}$$

where V_d^* is the optimal dual value; $y_j = \epsilon_j + \sum_i \ell_{ij}^\top x_i$; $\mathcal{C}$ stands for the constraints $A_i x_i = b_i$, $x_i \in X_i$, $\forall i$; and $f_j^*(\lambda_j) = \max_z\{\lambda_j z - f_j(z)\}$ is the Lagrangian dual of f_j. (For hinge loss functions, $f_j^*(\lambda_j) = \lambda_j d_j + \mathbb{I}(0 \leq \lambda_j \leq k)$, where $\mathbb{I}(\mathsf{test})$ is 0 or ∞ according to whether test is satisfied or not.) Thus, the two terms in Eq. 12 are: (1) a penalty to prevent resource prices λ_j from rising too high, and (2) the sum of agent utilities, including costs based on λ_j for resource usage.

A. Decoupling

Given a price vector λ, we can *decouple* the optimization of Eq. 12: each agent i optimizes its own dual value $V_d^i(\lambda)$ using the *slave problem*

$$V_d^i(\lambda) = \max_{x_i} \Big[c_i^\top x_i - \sum_{j=1}^m \lambda_j \Big(\epsilon_j + \sum_i \ell_{ij}^\top x_i\Big)\Big] \text{ s.t. } \mathcal{C}_i \tag{13}$$

where $\mathcal{C}_i$ is agent i's local constraints. The *master program* is now to compute the optimal dual value V_d^* using the following reformulation of Eq. 12:

$$V_d^* = \inf_\lambda V_d(\lambda) \tag{14}$$

$$V_d(\lambda) = \sum_{j=1}^{m} f_j^*(\lambda_j) + \sum_{i=1}^{n} V_d^i(\lambda) \tag{15}$$

Here, $V_d(\lambda)$ is the dual value for a specific λ.

B. Subgradient Descent

We can solve Eq. 14 by iteratively updating the price vector λ using projected subgradient descent. A subgradient of V_d is defined as any function $\partial[V_d(\lambda)]$ satisfying $V_d(\lambda') \geq V_d(\lambda) + (\lambda' - \lambda)^\top \partial[V_d(\lambda)]$ for all λ' and λ. The projected subgradient descent algorithm iteratively updates the old price vector λ to the new price vector λ' using:

$$\lambda' \leftarrow \Pi(\lambda - \eta_t \partial[V_d(\lambda)])$$

where η_t is a decreasing sequence of learning rates, and $\Pi(\lambda)$ is the greedy projection of λ onto the domain of V_d [6], [7]. We then use the following commonly-used lemma to compute $\partial[V_d(\lambda)]$:

Lemma 1: For any function $g(\lambda) = \sup_{x \in X}[\lambda^\top(Qx + r) - s(x)]$, if $x^* \in \arg\max_{x \in X}[\lambda^\top(Qx + r) - s(x)]$ for some λ, then $\partial[g(\lambda)] = Qx^* + r$.
To use the lemma, we reformulate Eq. 15 as

$$\begin{aligned} V_d(\lambda) = &\sum_{j=1}^{m} \max_z[\lambda_j z - f_j(z)] + \\ &\sum_{i=1}^{n} \max_{x_i} \left[c_i^\top x_i - \sum_{j=1}^{m} \lambda_j \ell_{ij}^\top x_i \right] \text{ s.t. } \mathcal{C} \end{aligned} \tag{16}$$

Thus, if we define $z_j^* = \arg\max_z[\lambda_j z - f_j(z)]$ and $x_i^* = \arg\max_{x_i \text{ s.t. } \mathcal{C}_i}[c_i^\top x_i - \sum_{j=1}^{m} \lambda_j \ell_{ij}^\top x_i]$, then

$$\partial[V_d(\lambda_j)] = z_j^* - \sum_i \ell_{ij}^\top x_i^* \tag{17}$$

according to Lemma 1. For example, we can always take $z_j^* = d_j$ for hinge loss functions, so $\partial[V_d(\lambda_j)] = d_j - \sum_i \ell_{ij}^\top x_i^*$, which is the difference between the resource limit and usage.

To implement the projection Π, we also need to know the domain of $V_d(\lambda)$. The domain of V_d is determined by the domains of the dual functions f_j^*. Since we assumed that each f_j is piecewise linear, we just need $\alpha_j^{\min} \leq \lambda_j \leq \alpha_j^{\max}$, where $\alpha_j^{\min}$ and $\alpha_j^{\max}$ are the smallest and largest slopes of the linear pieces. So, to calculate $\Pi(\lambda)$, we just set $\lambda_j \leftarrow \max(\alpha_j^{\min}, \min(\alpha_j^{\max}, \lambda_j))$.

C. SLR Algorithm

Plugging the above subgradient and projection operator into the update rule yields our final algorithm, which we call *Subgradient Lagrangian Relaxation* (SLR). Fig. 2 shows the pseudocode, where $\eta > 0$ is an initial learning rate and T is the desired maximum number of iterations. The outputs are an average policy for each agent ($\bar{x}_i$) as well as an average price for each resource ($\bar{\lambda}_j$).

Contrary to existing mechanisms for Lagrangian relaxation, a major difference is that the algorithm can be stopped if the dual value $V_d(\bar{\lambda})$ (Eq. 16) is close to the primal value $V_p(\bar{x})$ (Eq. 7). (Here $\bar{\lambda}$ is the vector of λ_j for all j, and $\bar{x}$ is the vector of $\bar{x}_i$ for all i, calculated on the last two lines of pseudocode.) This convergence criterion not only allows for performance bounds (see the next section) but also prevents the oscillation observed in problems with homogeneous agents.

We cannot directly execute the final aggregated policy $\bar{x}$: since the domains X_i are typically non-convex, averaging feasible solutions (as in the pseudocode) does not typically yield a feasible solution. We propose a simple remedy for this problem: each agent should independently pick a random locally-feasible policy $x_i \in X_i$ according to a distribution which makes $\mathbb{E}(x_i) = \bar{x}_i$. (One such distribution, perhaps the simplest and most general, is the uniform distribution over x_{it} for $t = 1 \ldots T$; but, other domain-specific distributions may be preferable.) However, the resulting global policy $\hat{x}$ might have poor performance, or might not even be a feasible solution, due to the coupling constraints. We remedy this problem by increasing ϵ_j (described in Eq. 10) sufficiently large such that the probability of $\hat{x}$ being a near-optimal solution is reasonably high (see proof below).

Mapping our illustrative theme park crowd management to SLR, each patron p corresponds to an agent i; for each patron p, the constants ${}_pc_{b,a}^t$ for all buckets b and attractions a together corresponds to the vector c_i of that agent, and the variable ${}_px_{b,a}^t$ for each bucket b and attraction a corresponds to a variable x_{it} for the current iteration t of that agent. Prices λ correspond to the global constraints (2), and keep each bucket of each queue from getting too crowded.

It should be noted that SLR is applicable to cases where there is uncertainty associated with coupling constraints. An example of such an uncertainty would be in problems where the capacity of a resource is unknown at planning time. This is handled by executing SLR over different samples of the uncertainty [8].

VI. Theoretical Results

The analysis of SLR can be divided into two parts: first, demonstrating that we can solve the Lagrangian relaxation of the global planning problem, and second, demonstrating that our randomized selection rule doesn't reduce our payoff too much compared to the relaxed solution.

For the first part, we can measure the performance of the subgradient iteration based on the *duality gap*, which is the difference between the primal and dual solution values $V_p(\bar{x})$ and $V_d(\bar{\lambda})$.

The dual value $V_d(\lambda)$ can be found from Eq. 16. If $\bar{x}_i$ is the current average solution for agent i and $\bar{y}_j = \sum_i \ell_{ij}^\top \bar{x}_i$,

```
Inputs: $c_i, \mathcal{C}_i, \ell_{ij}, f_j, \eta, T$        Outputs: $\bar{x}_i, \bar{\lambda}_j$
$\lambda_{j0} \leftarrow 0$                                              $j = 1 \dots m$
for $t \leftarrow 1, 2, \dots, T$
    $x_{it} \leftarrow \arg\max_{x_i \text{ s.t. } \mathcal{C}} [c_i^\top x_i - \sum_{j=1}^m \lambda_j \ell_{ij}^\top x_i]$    $i = 1 \dots n$
    $y_j \leftarrow \epsilon_j + \sum_{i=1}^n \ell_{ij}^\top x_{it}$                          $j = 1 \dots m$
    $z_j \leftarrow \arg\max_z [\lambda_j z - f_j(z)]$                      $j = 1 \dots m$
    $\lambda_{jt} \leftarrow \lambda_{j,t-1} + \frac{\eta}{\sqrt{t}}(y_j - z_j)$                      $j = 1 \dots m$
    $\lambda_{jt} \leftarrow \max(\alpha_j^{\min}, \min(\alpha_j^{\max}, \lambda_{jt}))$          $j = 1 \dots m$
    $\bar{x}_i \leftarrow \frac{1}{t} \sum_{k=1}^t x_{ik}$                              $i = 1 \dots n$
    $\bar{\lambda}_j \leftarrow \frac{1}{t} \sum_{k=1}^t \lambda_{jk}$                              $j = 1 \dots m$
```

Figure 2. Subgradient Lagrangian Relaxation pseudocode (see text for descriptions of inputs and outputs)

then the primal value is $V_p(\bar{x}) = \sum_i c_i^\top \bar{x}_i$.

Standard results about subgradient descent [9] imply that, after T iterations, the difference between the smallest dual value and the largest primal value encountered so far is $O(1/\sqrt{T})$. So, if $\bar{x}$ were guaranteed to be feasible, we would be done: we would simply need to set T large enough so that $V_d(\bar{\lambda}) - V_p(\bar{x})$ is smaller than some error tolerance. This is the convergence criterion that we have used in SLR.

Unfortunately, the domains X_i are typically non-convex, and $\bar{x}$ can be infeasible. So, we need the second part of our analysis: showing that our randomized selection rule extracts a near-optimal feasible solution from the average solution $\bar{x}$. Lemma 2 and Theorem 1 do so.

Intuitively, Theorem 1 tells us how SLR scales as the number of agents n increases. It lets us choose our parameters (resource slack t, convergence tolerance γ for subgradient descent, and failure probability δ for a single attempt at rounding) so that we get the best possible guarantee on the performance of the final global plan; and, it tells us that, with these parameters, the performance approaches optimal as $n \to \infty$. Furthermore, with these parameters, the expected runtime of SLR will be a low-order polynomial in the problem size.

In more detail, suppose for concreteness that we scale our problem so that the optimal value V_p^* and optimal resource consumption $|y_j^*|$ remain approximately constant as n increases. Then we can pick parameters as described in Lemma 3 so that (1) the runtime of subgradient descent is a low-order polynomial; (2) the expected number of rounding attempts is constant (so the total expected runtime is still a low-order polynomial); and (3) the suboptimality bound goes to zero as $n \to \infty$. SLR is therefore a polynomial-time algorithm which solves large planning problems near-optimally, with an error that *decreases* as we move to larger problems.

Thus, SLR as a whole runs in polynomial time and makes a polynomial number of calls to the local planning subroutines. If SLR succeeds, it finds a solution whose quality is close to V_p^* (with an error that decreases as $O(\frac{1}{\sqrt{n}})$). Each randomization of the final policy x has a constant failure probability, so we can achieve any desired failure probability by trying several independent randomizations and taking the best (increasing runtime by at most a constant factor). SLR is therefore a polynomial-time algorithm which solves large planning problems near-optimally, with an error that *decreases* as we move to larger problems.

Lemma 2: If each agent has a bounded influence on resource use and overall utility (Eqs. 8–9), and randomizes independently with $\mathbb{E}(x_i) = \bar{x}_i$, then with high probability, $\sum_i c_i^\top x_i$ is close to $\sum_i c_i^\top \bar{x}_i$ and $\sum_i \ell_{ij} x_i$ is close to $\sum_i \ell_{ij} \bar{x}_i$. In particular, for any t, set

$$\delta = e^{-nt^2/2U^2|V_p^*|^2} + m e^{-nt^2/2U^2|y_j^*|^2}$$

with probability at least $1 - \delta$, the following holds.

$$\sum_i c_i^\top x_i \geq \sum_i c_i^\top \bar{x}_i - t$$

$$\sum_i \ell_{ij} x_i \leq \sum_i \ell_{ij} \bar{x}_i + t \qquad \forall j$$

Proof: By the Hoeffding-Azuma inequality and Eq. 8,

$$\mathbb{P}[\textstyle\sum_i c_i^\top x_i < \sum_i c_i^\top \bar{x}_i - t] \leq e^{-nt^2/2U^2|V_p^*|^2}$$

Similarly, by Hoeffding-Azuma and Eq. 9, for each j,

$$\mathbb{P}[\textstyle\sum_i \ell_{ij}^\top x_i > \sum_i \ell_{ij}^\top \bar{x}_i + t] \leq e^{-nt^2/2U^2|y_j^*|^2}$$

A union bound then yields the desired result. ■

Theorem 1: Suppose influence limits are guaranteed by Eqs. 8–11. Fix $t \leq \epsilon_{\max}$, set $\epsilon_j = t$ for all j, and run SLR (Fig. 2) to some tolerance γ. Let each agent randomize independently with $\mathbb{E}(x_i) = \bar{x}_i$. Set

$$\delta = e^{-nt^2/2U^2|V_p^*|^2} + m e^{-nt^2/2U^2|y_j^*|^2}$$

Then with probability at least $1 - \delta$,

$$V_p(x) \geq V_p^* - \Delta/n - (\kappa m + 1)t - \gamma$$

Proof: By Eq. 11, setting $\epsilon_j = t$ means that the overall value of the modified program is $V_\epsilon^* \geq V_p^* - \Delta/n - \kappa m t$. Running SLR to tolerance γ will yield a relaxed solution $\bar{x}$ with value $V_{\text{relax}} \geq V_\epsilon^* - \gamma$ in the modified program. Split $V_{\text{relax}} = \sum_i c_i^\top \bar{x}_i - \sum_j f_j(t + \sum_i \ell_{ij}^\top \bar{x}_i)$. By Lemma 2, with probability at least $1 - \delta$, the randomly selected solution x will have

$$\sum_i c_i^\top x_i \geq \sum_i c_i^\top \bar{x}_i - t$$

and, for all j,

$$\sum_i \ell_{ij} x_i \leq \sum_i \ell_{ij} \bar{x}_i + t$$

which implies

$$f_j(\sum_i \ell_{ij} x_i) \leq f_j(\sum_i \ell_{ij} \bar{x}_i + t)$$

since f_j is monotone increasing. So, with probability at least $1-\delta$,

$$\sum_i c_i^\top x_i - \sum_j f_j(\sum_i \ell_{ij} x_i) \geq V_{\text{relax}} - t$$

Since the left-hand side above is $V_p(x)$, substituting in $V_{\text{relax}} \geq V_\epsilon^* - \gamma$ and $V_\epsilon^* \geq V_p^* - \Delta/n - \kappa m t$ yields the desired bound. ■

Lemma 3: We can pick parameters t, γ, and δ to satisfy properties (1)–(3) from above.

Proof: For (2), if we keep the failure probability δ bounded for each attempt at rounding, we need an expected-constant number of attempts before success. We can do so by picking t large enough: e.g., we can pick $t = \Omega(\frac{\ln 1/\delta}{\sqrt{n}})$ to achieve any desired failure probability δ, say $\delta = 0.01$. For (3), in the bound on $V_p(x)$, the first term (V_p^*)will remain constant as $n \to \infty$. The second term (Δ/n) will go to zero as $n \to \infty$. The third term will also go to zero, since $t \to 0$ as $n \to \infty$. The fourth term (the convergence tolerance γ) is under our control: e.g., we can take $\gamma = \frac{1}{\sqrt{n}}$ so $\gamma \to 0$ as $n \to \infty$. For (1), the number of iterations of SLR depends polynomially on $1/\gamma$ (and therefore on n); the time for each iteration of SLR depends polynomially on n and m; and each iteration makes polynomially many calls to the local planning subroutines. ■

VII. Experimental Results

In this section, we show that SLR is able to find near-optimal solutions for large-scale problems using the illustrative theme park crowd management domain. Using the notation from the "Illustrative Domain" section, we categorize each problem with the following parameters: (a) horizon H; (b) number of attractions $|A|$; (c) service rates δ_{a_i}; (d) number of patrons n; and (e) number of distinct patron preferences k. In our experiments, we fix $|A|$ to 10. We vary δ_{a_i} from 5 to 15 for all attractions, H from 5 to 10, n from 500 to 1500, and k from 5 to 10. These parameter ranges imply that the smallest problem has 500 agents (= n) and 250 resources (= $H^2 \times |A|$) and the largest problem has 1500 agents and 1000 resources. We run our experiments on a quad-core Intel Xeon 3.16GHz CPU with 16GB memory. Fig. 3 shows a set of representative results, where we plot the primal and dual values across iterations. We make the following observations:

- Fig. 3(a) shows that as H increases, the converged solution improves as well. Intuitively, the reason is that the optimal solution quality increases with the horizon – some patrons are now able to visit unvisited attractions that they would otherwise have skipped if H was smaller.

Table I
Solution Quality of the Centralized MIP and SLR

n	Centralized MIP	TREMOR		SLR	
5	100	100	[0.00]	100	[0.00]
10	200	200	[0.00]	200	[0.00]
25	500	500	[0.00]	499	[0.20]
50	950	945	[0.00]	944	[0.63]
75	1150	1125	[2.17]	1138	[1.04]
100	1150	1085	[5.65]	1138	[1.04]

- Fig. 3(b) shows that as n and δ_{a_i} increase, the converged solution improves as well. Intuitively, the reason is that the optimal solution quality increases with the number of patrons and service rates – the number of patrons serviced increases with these two parameters.
- Fig. 3(c) shows that as k increases, the converged solution remains very similar. Intuitively, the reason is that the optimal solution quality likely remained unchanged – the maximum number of patrons that can be serviced is already reached. The maximum number of patrons that can be serviced is 500 (= $|A| \times \delta_{a_i} \times H$), which is less than n.
- All three figures show that the duality gap, which reflects the error bound, is very small upon convergence indicating that the solution is near optimal.

Additionally, we compare SLR with the centralized MIP described in the "Illustrative Domain" section as well as TREMOR [10] (a DEC-POMDP algorithm that uses model shaping) on smaller sandbox problems to quantitatively gauge the actual error. For these problems, we fix $|A|$ to 5, δ_{a_i} to 5 for all attractions, H to 5 and k to 5. We vary n from 5 to 100. Table I shows our results, where the numbers in brackets indicate the percentage error. Unfortunately, we could not compare our approach on larger problems as the centralized MIP failed to converge in CPLEX after one hour even for our smallest problem. In contrast, SLR converges to a solution after about 1000-4000 iterations, where each iteration takes about 0.5s. In these problems, TREMOR was able to find solutions of similar quality as those found by SLR. However, TREMOR does not provide any quality guarantees on the solutions found unlike SLR. It is also possible to construct specific problems where TREMOR will perform arbitrarily bad.

VIII. Related Work

Our crowd management problem is also very similar to Prize Collecting TSPs (PC-TSP) [11], where an agent plans a path through the theme park to collect "prizes" at each attraction. However, PC-TSPs are not directly applicable: PC-TSP is a single-agent problem, while our crowd management problem is a multi-agent problem.

Lagrangian relaxation has been used extensively in solving large-scale optimization problems [3] with most of the work done on designing "good" subgradient descent formulas to achieve quick global convergence [12], [13] as

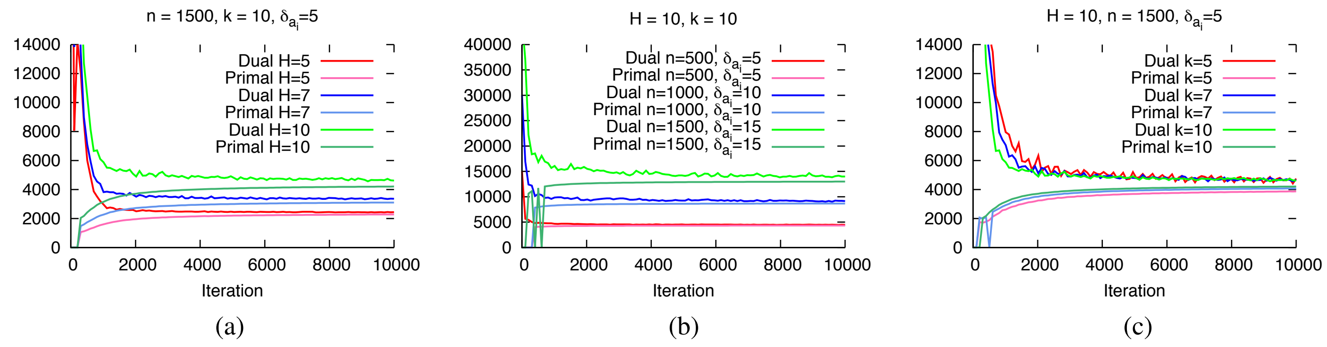

Figure 3. Experimental Results of SLR

well as adapting standard Lagrangian relaxation to handle oscillations caused by agent homogeneity [14]. Unfortunately, the ideas on oscillation in the papers above do not scale well, as the slave problem for each homogeneous agent needs to be solved in sequence.

As far as multi-agent planning is concerned, researchers have previously used Lagrangian relaxation with some success; however, past work often either assumes infinitely divisible resources [15], [4], or tries to get an exact solution and therefore doesn't scale to really large problems [16]. Researchers have also studied restrictions to the type of planning problem considered, in some cases providing strong approximation guarantees using polynomial-time algorithms [17], [1]. In contrast, our work allows near-arbitrary local planning problems for the individual agents, and a very general mechanism for agent interaction. Instead of restricting the problem type, we gain leverage from the assumption of bounded influence for any individual agent.

There has been research in weakly coupled MDPs [18], [19] where resource coupled interactions have been considered. However, the approaches have either focused on a simpler problem (sequential dependence on resources) or have considered optimal approaches that are not scalable. Researchers have also proposed heuristic model-shaping approaches for solving a sub-class of DEC-POMDPs [10]. While their approach is applicable and can scale to the problems of interest in this paper, they do not have guarantees on convergence and final solution quality. In fact, even on small problems, we were able to demonstrate significant difference in solution quality.

Acknowledgments

This research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office.

References

[1] S. Koenig, P. Keskinocak, and C. Tovey, "Progress on agent coordination with cooperative auctions [Senior Member Paper]," in *Proceedings of AAAI*, 2010, pp. 1713–1717.

[2] P. Varakantham, S.-F. Cheng, and T. D. Nguyen, "Decentralized decision support for an agent population in dynamic and uncertain domains," in *Proceedings of AAMAS*, 2011, pp. 1147–1148.

[3] M. Fisher, "An applications oriented guide to Lagrangian relaxation," *Interfaces*, vol. 15, no. 2, pp. 10–21, 1985.

[4] C. Guestrin and G. Gordon, "Distributed planning in hierarchical factored MDPs," in *Proceedings of UAI*, 2002, pp. 197–206.

[5] D. Dolgov, M. James, and M. Samples, "Combinatorial resource scheduling for multiagent MDPs," in *Proceedings of AAMAS*, 2007, pp. 657–664.

[6] G. Gordon, "Regret bounds for prediction problems," in *Proceedings of COLT*, 1999, pp. 29–40.

[7] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proceedings of ICML*, 2003, pp. 928–936.

[8] A. Ng and M. Jordan, "PEGASUS: A policy search method for large MDPs and POMDPs," in *Proceedings of UAI*, 2000, pp. 406–415.

[9] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[10] P. Velagapudi, P. Varakantham, P. Scerri, and K. Sycara, "Distributed model shaping for scaling to decentralized POMDPs with hundreds of agents," in *Proceedings of AAMAS*, 2011, pp. 955–962.

[11] E. Balas, "The prize collecting traveling salesman problem," *Networks*, vol. 19, pp. 621–636, 1989.

[12] M. Held, P. Wolfe, and H. Crowder, "Validation of subgradient optimization," *Mathematical Programming*, vol. 6, no. 1, pp. 62–88, 1974.

[13] C. Kaskavelis and M. Caramanis, "Efficient Lagrangian relaxation algorithms for industry size job-shop scheduling problems," *IIE Transactions*, vol. 30, no. 11, pp. 1085–1097, 1998.

[14] Q. Zhai, X. Guan, and J. Cui, “Unit commitment with identical units successive subproblem solving method based on Lagrangian relaxation,” *IEEE Transactions on Power Systems*, vol. 17, no. 4, pp. 1250–1257, 2002.

[15] J.-P. Callies and G. Gordon, “No-regret learning and a mechanism for distributed multiagent planning,” in *Proceedings of AAMAS*, 2008, pp. 509–516.

[16] S. A. Hong and G. Gordon, “Optimal distributed market-based planning for multi-agent systems with shared resources,” in *Proceedings of AISTATS*, 2011, pp. 351–360.

[17] S. Koenig, C. Tovey, M. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, A. Meyerson, and S. Jain, “The power of sequential single-item auctions for agent coordination [Nectar Paper],” in *Proceedings of AAAI*, 2006, pp. 1625–1629.

[18] S. Singh and D. Cohn, “How to dynamically merge markov decision processes,” in *Proceedings of NIPS*, 1998, pp. 1057–1063.

[19] N. Meuleau, M. Hauskrecht, K. E. Kim, L. Peshkin, L. Kaelbling, T. Dean, and C. Boutilier, “Solving very large weakly coupled Markov decision processes,” in *Proceedings of AAAI*, 1998, pp. 165–172.